3 과목 데이터베이스 구축

핵심 146 데이터베이스 설계의 개념 및 고려사항

데이터베이스 설계란 사용자의 요구를 분석하여 그것들을 컴퓨터에 저장할 수 있는 데이터베이스의 구조에 맞게 변형한 후 특정 DBMS로 데이터베이스를 구현하여 일반 사용자들이 사용하게 하는 것이다.

데이터베이스 설계 시 고려사항

무결성	삽입, 삭제, 갱신 등의 연산 후에도 데이터베이스에 저장된 데이터가 정해진 제약 조건을 항상 만족해야 한다.
일관성	데이터베이스에 저장된 데이터들 사이나, 특정 질의에 대한 응답이 처음부터 끝까지 변함없이 일정해야 한다.
회복	시스템에 장애가 발생했을 때 장애 발생 직전의 상태로 복구할 수 있어야 한다.
보안	불법적인 데이터의 노출 또는 변경이나 손실로부터 보호할 수 있어야 한다.
효율성	응답시간의 단축, 시스템의 생산성, 저장 공간의 최적화 등이 가능해야 한다.
데이터베이스 확장	데이터베이스 운영에 영향을 주지 않으면서 지속적으로 데이터를 추가할 수 있어야 한다.

핵심 147 데이터베이스 설계 순서

요구 조건 분석	요구 조건 명세서 작성
↓	
개념적 설계	개념 스키마, 트랜잭션 모델링, E-R 모델
↓	
논리적 설계	목표 DBMS에 맞는 논리 스키마 설계, 트랜잭션 인터페이스 설계
↓	
물리적 설계	목표 DBMS에 맞는 물리적 구조의 데이터로 변환
↓	
구현	목표 DBMS의 DDL(데이터 정의어)로 데이터베이스 생성, 트랜잭션 작성

핵심 148 개념적 설계(정보 모델링, 개념화)

개념적 설계란 정보의 구조를 얻기 위하여 현실 세계의 무한성과 계속성을 이해하고, 다른 사람과 통신하기 위하여 현실 세계에 대한 인식을 추상적 개념으로 표현하는 과정이다.

- 개념적 설계 단계에서는 개념 스키마 모델링과 트랜잭션 모델링을 병행 수행한다.
- 개념적 설계 단계에서는 요구 분석 단계에서 나온 결과인 요구 조건 명세를 DBMS에 독립적인 E-R 다이어그램으로 작성한다.
- DBMS에 독립적인 개념 스키마를 설계한다.

핵심 149 논리적 설계(데이터 모델링)

논리적 설계 단계란 현실 세계에서 발생하는 자료를 컴퓨터가 이해하고 처리할 수 있는 물리적 저장장치에 저장할 수 있도록 변환하기 위해 특정 DBMS가 지원하는 논리적 자료 구조로 변환(mapping)시키는 과정이다.

- 개념 세계의 데이터를 필드로 기술된 데이터 타입과 이 데이터 타입들 간의 관계로 표현되는 논리적 구조의 데이터로 모델화한다.
- 개념적 설계가 개념 스키마를 설계하는 단계라면 논리적 설계에서는 개념 스키마를 평가 및 정제하고 DBMS에 따라 서로 다른 논리적 스키마를 설계하는 단계이다.
- 트랜잭션의 인터페이스를 설계한다.
- 관계형 데이터베이스라면 테이블을 설계하는 단계이다.

핵심 150 물리적 설계(데이터 구조화)

물리적 설계란 논리적 설계 단계에서 논리적 구조로 표현된 데이터를 디스크 등의 물리적 저장장치에 저장할 수 있는 물리적 구조의 데이터로 변환하는 과정이다.

- 물리적 설계 단계에서는 다양한 데이터베이스 응용에 대해 처리 성능을 얻기 위해 데이터베이스 파일의 저장 구조 및 액세스 경로를 결정한다.
- 저장 레코드의 형식, 순서, 접근 경로, 조회가 집중되는 레코드와 같은 정보를 사용하여 데이터가 컴퓨터에 저장되는 방법을 묘사한다.

핵심 151 데이터 모델

데이터 모델은 현실 세계의 정보들을 컴퓨터에 표현하기 위해서 단순화, 추상화하여 체계적으로 표현한 개념적 모형이다.

- 데이터 모델은 데이터, 데이터의 관계, 데이터의 의미 및 일관성, 제약 조건 등을 기술하기 위한 개념적 도구들의 모임이다.
- 현실 세계를 데이터베이스에 표현하는 중간 과정, 즉 데이터베이스 설계 과정에서 데이터의 구조(Schema)를 논리적으로 표현하기 위해 사용되는 지능적 도구이다.

데이터 모델의 구성 요소

개체(Entity)	데이터베이스에 표현하려는 것으로, 사람이 생각하는 개념이나 정보 단위 같은 현실 세계의 대상체이다.
속성(Attribute)	데이터의 가장 작은 논리적 단위로서 파일 구조상의 데이터 항목 또는 데이터 필드에 해당한다.
관계(Relationship)	개체 간의 관계 또는 속성 간의 논리적인 연결을 의미한다.

데이터 모델에 표시할 요소

구조(Structure)	논리적으로 표현된 개체 타입들 간의 관계로서 데이터 구조 및 정적 성질을 표현한다.
연산(Operation)	데이터베이스에 저장된 실제 데이터를 처리하는 작업에 대한 명세로서 데이터베이스를 조작하는 기본 도구이다.
제약 조건(Constraint)	데이터베이스에 저장될 수 있는 실제 데이터의 논리적인 제약 조건이다.

핵심 152 개체(Entity)

개체의 정의 및 특징

개체(Entity)는 데이터베이스에 표현하려는 것으로, 사람이 생각하는 개념이나 정보 단위 같은 현실 세계의 대상체이다.

- 개체는 실세계에 독립적으로 존재하는 유형, 무형의 정보로서 서로 연관된 몇 개의 속성으로 구성된다.
- 유일한 식별자(Unique Identifier)에 의해 식별이 가능하다.
- 다른 개체와 하나 이상의 관계(Relationship)가 있다.

개체 선정 방법

- 업무 분석에 관한 내용을 구체적으로 설명한 업무 기술서를 이용한다.
- 실제 업무를 담당하고 있는 담당자와 인터뷰를 한다.
- 업무 기술서와 인터뷰에서 확인하지 못한 정보가 있는지 실제 업무를 직접 견학하여 확인한다.
- 실제 업무에 사용되고 있는 장부와 전표를 이용한다.
- 이미 구축된 시스템이 있는 경우 해당 시스템의 산출물을 검토한다.

개체명 지정 방법

- 일반적으로 해당 업무에서 사용하는 용어로 지정한다.
- 약어 사용은 되도록 제한한다.
- 가능하면 단수 명사를 사용한다.
- 모든 개체명은 유일해야 한다.
- 가능하면 개체가 생성되는 의미에 따라 이름을 부여한다.

핵심 153 속성의 정의 및 특징

- 속성(Attribute)은 데이터베이스를 구성하는 가장 작은 논리적 단위이다.
- 파일 구조상의 데이터 항목 또는 데이터 필드에 해당한다.
- 속성은 개체를 구성하는 항목이다.
- 속성은 개체의 특성을 기술한다.
- 속성의 수를 디그리(Degree) 또는 차수라고 한다.

핵심 154 속성의 종류

속성의 특성에 따른 분류

기본 속성 (Basic Attribute)	• 업무 분석을 통해 정의한 속성이다. • 속성 중 가장 많고 일반적이다. • 업무로부터 분석한 속성이라도 업무상 코드로 정의한 속성은 기본 속성에서 제외된다.
설계 속성 (Designed Attribute)	• 원래 업무상 존재하지 않고 설계 과정에서 도출해내는 속성이다. • 업무에 필요한 데이터 외에 데이터 모델링을 위해 업무를 규칙화하려고 속성을 새로 만들거나 변형하여 정의하는 속성이다.

파생 속성 (Derived Attribute)	• 다른 속성으로부터 계산이나 변형 등의 영향을 받아 발생하는 속성이다. • 파생 속성은 되도록 적은 수를 정의하는 것이 좋다.

개체 구성 방식에 따른 분류

기본키 속성(Primary Key Attribute)	개체를 식별할 수 있는 속성이다.
외래키 속성(Foreign Key Attribute)	다른 개체와의 관계에서 포함된 속성이다.
일반 속성	개체에 포함되어 있고 기본키, 외래키에 포함되지 않은 속성이다.

핵심 155 속성명 지정 원칙

- 속성명은 웹이나 클라이언트/서버(Client/Server) 등 어떠한 환경에서든 사용자 인터페이스에 나타나기 때문에 정확하고 혼란이 없도록 지정해야 한다.
- 속성명 지정 원칙
  - 해당 업무에서 사용하는 용어로 지정한다.
  - 서술형으로 지정하지 않는다.
  - 가급적이면 약어의 사용은 제한한다.
  - 개체명은 속성명으로 사용할 수 없다.
  - 개체에서 유일하게 식별 가능하도록 지정한다.

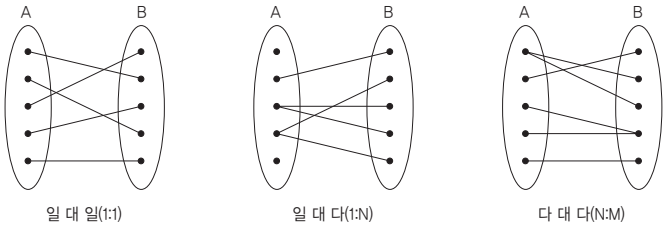
핵심 156 관계(Relationship)

관계는 개체와 개체 사이의 논리적인 연결을 의미한다.

- 관계에는 개체 간의 관계와 속성 간의 관계가 있다.

관계의 형태

일 대 일(1:1)	개체 집합 A의 각 원소가 개체 집합 B의 원소 한 개와 대응하는 관계이다.
일 대 다(1:N)	개체 집합 A의 각 원소는 개체 집합 B의 원소 여러 개와 대응하고 있지만, 개체 집합 B의 각 원소는 개체 집합 A의 원소 한 개와 대응하는 관계이다.
다 대 다(N:M)	개체 집합 A의 각 원소는 개체 집합 B의 원소 여러 개와 대응하고, 개체 집합 B의 각 원소도 개체 집합 A의 원소 여러 개와 대응하는 관계이다.



일 대 일(1:1) 일 대 다(1:N) 다 대 다(N:M)

관계의 종류

종속 관계 (Dependent Relationship)	두 개체 사이의 주 · 종 관계를 표현한 것으로, 식별 관계와 비식별 관계가 있다.
중복 관계 (Redundant Relationship)	두 개체 사이에 2번 이상의 종속 관계가 발생하는 관계이다.
재귀 관계 (Recursive Relationship)	개체가 자기 자신과 관계를 갖는 것으로, 순환 관계(Recursive Relationship)라고도 한다.
배타 관계 (Exclusive Relationship)	• 개체의 속성이나 구분자를 기준으로 개체의 특성을 분할하는 관계로, 배타 AND 관계와 배타 OR 관계로 구분한다. • 배타 AND 관계는 하위 개체들 중 속성이나 구분자 조건에 따라 하나의 개체만을 선택할 수 있고, 배타 OR 관계는 하나 이상의 개체를 선택할 수 있다.

핵심 157 식별자(Identifier)

분류	식별자
대표성 여부	• 주 식별자(Primary Identifier) : 개체를 대표하는 유일한 식별자 • 보조 식별자(Alternate Identifier) : 주 식별자를 대신하여 개체를 식별할 수 있는 속성
스스로 생성 여부	• 내부 식별자(Internal Identifier) : 개체 내에서 스스로 만들어지는 식별자 • 외부 식별자(Foreign Identifier) : 다른 개체와의 관계(Relationship)에 의해 외부 개체의 식별자를 가져와 사용하는 식별자
단일 속성 여부	• 단일 식별자(Single Identifier) : 주 식별자가 한 가지 속성으로만 구성된 식별자 • 복합 식별자(Composit Identifier) : 주 식별자가 두 개 이상의 속성으로 구성된 식별자
대체 여부	• 원조 식별자(Original Identifier) : 업무에 의해 만들어지는 가공되지 않은 원래의 식별자로, 본질 식별자라고도 함 • 대리 식별자(Surrogate Identifier) : 주 식별자의 속성이 두 개 이상인 경우 속성들을 하나의 속성으로 묶어 사용하는 식별자로, 인조 식별자라고도 함

※ 주 식별자의 특징

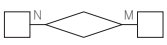
특징	내용
유일성	주 식별자에 의해 개체 내에 모든 인스턴스들이 유일하게 구분되어야 함
최소성	주 식별자를 구성하는 속성의 수는 유일성을 만족하는 최소 수가 되어야 함
불변성	주 식별자가 한 번 특정 개체에 지정되면 그 식별자는 변하지 않아야 함
존재성	주 식별자가 지정되면 식별자 속성에 반드시 데이터 값이 존재해야 함

핵심 158 E-R 모델의 개요

E-R 모델은 개념적 데이터 모델의 가장 대표적인 것으로, 1976년 피터 첸(Peter Chen)에 의해 제안되고 기본적인 구성 요소가 정립되었다.

- E-R 모델은 개체 타입(Entity Type)과 이들 간의 관계 타입(Relationship Type)을 이용해 현실 세계를 개념적으로 표현한다.
- E-R 모델에서는 데이터를 개체(Entity), 관계(Relationship), 속성(Attribute)으로 묘사한다.
- E-R 모델은 특정 DBMS를 고려한 것은 아니다.
- E-R 다이어그램으로 표현하며, 1:1, 1:N, N:M 등의 관계 유형을 제한 없이 나타낼 수 있다.

핵심 159 E-R 다이어그램

기호	기호 이름	의미
	사각형	개체(Entity) 타입
	마름모	관계(Relationship) 타입
	타원	속성(Attribute)
	이중 타원	다중값 속성(복합 속성)
	밑줄 타원	기본키 속성
	복수 타원	복합 속성 예 성명은 성과 이름으로 구성
	관계	1:1, 1:N, N:M 등의 개체 간 관계에 대한 대응수를 선 위에 기술함
	선, 링크	개체 타입과 속성을 연결

핵심 160 관계형 데이터 모델

관계형 데이터 모델은 가장 널리 사용되는 데이터 모델로, 2차원적인 표(Table)를 이용해서 데이터 상호 관계를 정의하는 DB 구조를 말한다.

- 파일 구조처럼 구성한 테이블들을 하나의 DB로 묶어서 테이블 내에 있는 속성들 간의 관계(Relationship)를 설정하거나 테이블 간의 관계를 설정하여 이용한다.
- 기본키(Primary Key)와 이를 참조하는 외래키(Foreign Key)로 데이터 간의 관계를 표현한다.
- 계층 모델과 망 모델의 복잡한 구조를 단순화시킨 모델이다.
- 관계형 모델의 대표적인 언어는 SQL이다.
- 1:1, 1:N, N:M 관계를 자유롭게 표현할 수 있다.

핵심 161 관계형 데이터베이스의 Relation 구조

릴레이션은 데이터들을 표(Table)의 형태로 표현한 것으로 구조를 나타내는 릴레이션 스키마와 실제 값들인 릴레이션 인스턴스로 구성된다.

튜플(Tuple)

- 튜플은 릴레이션을 구성하는 각각의 행을 말한다.
- 튜플은 속성의 모임으로 구성된다.
- 파일 구조에서 레코드와 같은 의미이다.
- 튜플의 수를 카디널리티(Cardinality) 또는 기수, 대응수라고 한다.

속성(Attribute)

- 속성은 데이터베이스를 구성하는 가장 작은 논리적 단위이다.
- 파일 구조상의 데이터 항목 또는 데이터 필드에 해당된다.
- 속성은 개체의 특성을 기술한다.
- 속성의 수를 디그리(Degree) 또는 차수라고 한다.

도메인(Domain)

- 도메인은 하나의 애트리뷰트가 취할 수 있는 같은 타입의 원자(Atomic)값들의 집합이다.
- 도메인은 실제 애트리뷰트 값이 나타날 때 그 값의 합법 여부를 시스템이 검사하는데에도 이용된다.
  - 예 성별 애트리뷰트의 도메인은 '남'과 '여'로, 그 외의 값은 입력될 수 없다.

불합격 방지용 안전장치 기억상자

틀린 문제만 모아 오답 노트를 만들고 싶다고요?
까먹기 전에 다시 한 번 복습하고 싶다고요?
지금 당장 QR 코드를 스캔해 보세요.

핵심 162 릴레이션의 특징

- 한 릴레이션에는 똑같은 튜플이 포함될 수 없으므로 릴레이션에 포함된 튜플들은 모두 상이하다.
  - 예 〈학생〉 릴레이션을 구성하는 김예소 레코드는 김예소에 대한 학적 사항을 나타내는 것으로 〈학생〉 릴레이션 내에서는 유일하다.
- 한 릴레이션에 포함된 튜플 사이에는 순서가 없다.
  - 예 〈학생〉 릴레이션에서 김예소 레코드와 고강민 레코드의 위치가 바뀌어도 상관없다.
- 튜플들의 삽입, 삭제 등의 작업으로 인해 릴레이션은 시간에 따라 변한다.
  - 예 〈학생〉 릴레이션에 새로운 학생의 레코드를 삽입하거나 기존 학생에 대한 레코드를 삭제함으로써 테이블은 내용 면에서나 크기 면에서 변하게 된다.
- 릴레이션 스키마를 구성하는 속성들 간의 순서는 중요하지 않다.
  - 예 학번, 이름 등의 속성을 나열하는 순서가 이름, 학번 순으로 바뀌어도 데이터 처리에는 아무런 영향을 미치지 않는다.
- 속성의 유일한 식별을 위해 속성의 명칭은 유일해야 하지만, 속성을 구성하는 값은 동일한 값이 있을 수 있다.
  - 예 각 학생의 학년을 기술하는 속성인 '학년'은 다른 속성명들과 구분되어 유일해야 하지만 '학년' 속성에는 2, 1, 2, 4 등이 입력된 것처럼 동일한 값이 있을 수 있다.
- 릴레이션을 구성하는 튜플을 유일하게 식별하기 위해 속성들의 부분집합을 키(Key)로 설정한다.
  - 예 〈학생〉 릴레이션에서는 '학번'이나 '이름'이 튜플들을 구분하는 유일한 값인 키가 될 수 있다.
- 속성의 값은 논리적으로 더 이상 쪼갤 수 없는 원자값만을 저장한다.
  - 예 '학년'에 저장된 1, 2, 4 등은 더 이상 세분화할 수 없다.

핵심 163 키(Key)

키(Key)는 데이터베이스에서 조건에 만족하는 튜플을 찾거나 순서대로 정렬할 때 튜플들을 서로 구분할 수 있는 기준이 되는 애트리뷰트를 말한다.

후보키 (Candidate Key)	• 릴레이션을 구성하는 속성들 중에서 튜플을 유일하게 식별하기 위해 사용하는 속성들의 부분집합, 즉 기본키로 사용할 수 있는 속성들을 말한다. • 후보키는 릴레이션에 있는 모든 튜플에 대해서 유일성과 최소성을 만족시켜야 한다.
기본키 (Primary Key)	• 후보키 중에서 특별히 선정된 주키(Main Key)로 중복된 값을 가질 수 없다. • 한 릴레이션에서 특정 튜플을 유일하게 구별할 수 있는 속성이다. • 기본키는 NULL 값을 가질 수 없다. 즉 튜플에서 기본키로 설정된 속성에는 NULL 값이 있어서는 안 된다.
대체키 (Alternate Key)	• 후보키가 둘 이상일 때 기본키를 제외한 나머지 후보키를 의미한다. • 보조키라고도 한다.
슈퍼키 (Super Key)	• 한 릴레이션 내에 있는 속성들의 집합으로 구성된 키로서 릴레이션을 구성하는 모든 튜플들 중 슈퍼키로 구성된 속성의 집합과 동일한 값은 나타나지 않는다. • 슈퍼키는 릴레이션을 구성하는 모든 튜플에 대해 유일성은 만족시키지만, 최소성은 만족시키지 못한다.
외래키 (Foreign Key)	• 다른 릴레이션의 기본키를 참조하는 속성 또는 속성들의 집합을 의미한다. • 한 릴레이션에 속한 속성 A와 참조 릴레이션의 기본키인 B가 동일한 도메인 상에서 정의되었을 때의 속성 A를 외래키라고 한다.

핵심 164 무결성(Integrity)

무결성이란 데이터베이스에 저장된 데이터 값과 그것이 표현하는 현실 세계의 실제값이 일치하는 정확성을 의미한다.

개체 무결성(Entity Integrity, 실체 무결성)	기본 테이블의 기본키를 구성하는 어떤 속성도 Null 값이나 중복값을 가질 수 없다는 규정이다.
도메인 무결성(Domain Integrity, 영역 무결성)	주어진 속성 값이 정의된 도메인에 속한 값이어야 한다는 규정이다.
참조 무결성 (Referential Integrity)	외래키 값은 Null이거나 참조 릴레이션의 기본키 값과 동일해야 한다. 즉 릴레이션은 참조할 수 없는 외래키 값을 가질 수 없다는 규정이다.
사용자 정의 무결성 (User-Defined Integrity)	속성 값들이 사용자가 정의한 제약조건에 만족해야 한다는 규정이다.

핵심 165 관계대수의 개요

관계대수는 관계형 데이터베이스에서 원하는 정보와 그 정보를 검색하기 위해서 어떻게 유도하는가를 기술하는 절차적인 언어이다.

- 관계대수는 릴레이션을 처리하기 위해 연산자와 연산 규칙을 제공하는 언어로 피연산자가 릴레이션이고, 결과도 릴레이션이다.
- 질의에 대한 해를 구하기 위해 수행해야 할 연산의 순서를 명시한다.
- 관계대수에는 관계 데이터베이스에 적용하기 위해 특별히 개발한 순수 관계 연산자와 수학적 집합 이론에서 사용하는 일반 집합 연산자가 있다.
- 순수 관계 연산자
  - Select
  - Project
  - Join
  - Division
- 일반 집합 연산자
  - UNION(합집합)
  - INTERSECTION(교집합)
  - DIFFERENCE(차집합)
  - CARTESIAN PRODUCT(교차곱)

핵심 166 순수 관계 연산자

Select	릴레이션에 존재하는 튜플 중에서 선택 조건을 만족하는 튜플의 부분집합을 구하여 새로운 릴레이션을 만드는 연산이다. • 릴레이션의 행(가로)에 해당하는 튜플을 구하는 것이므로 수평 연산이라고도 한다. • 연산자의 기호는 그리스 문자 시그마(σ)를 사용한다.
Project	주어진 릴레이션에서 속성 리스트(Attribute List)에 제시된 속성 값만을 추출하여 새로운 릴레이션을 만드는 연산이다. 단 연산 결과에 중복이 발생하면 중복이 제거된다. • 릴레이션의 열(세로)에 해당하는 Attribute를 추출하는 것이므로 수직 연산자라고도 한다. • 연산자의 기호는 그리스 문자 파이(π)를 사용한다.
Join	공통 속성을 중심으로 두 개의 릴레이션을 하나로 합쳐서 새로운 릴레이션을 만드는 연산이다. • 연산자의 기호는 ⋈를 사용한다.
Division	X⊃Y인 두 개의 릴레이션 R(X)와 S(Y)가 있을 때, R의 속성이 S의 속성값을 모두 가진 튜플에서 S가 가진 속성을 제외한 속성만을 구하는 연산이다. • 연산자의 기호는 ÷를 사용한다.

핵심 167 관계해석(Relational Calculus)

관계해석은 관계 데이터 모델의 제안자인 코드(E. F. Codd)가 수학의 Predicate Calculus(술어 해석)에 기반을 두고 관계 데이터베이스를 위해 제안했다.

- 관계해석은 관계 데이터의 연산을 표현하는 방법으로, 원하는 정보를 정의할 때는 계산 수식을 사용한다.
- 관계해석은 원하는 정보가 무엇이라는 것만 정의하는 비절차적 특성을 지닌다.
- 튜플 관계해석과 도메인 관계해석이 있다.
- 기본적으로 관계해석과 관계대수는 관계 데이터베이스를 처리하는 기능과 능력면에서 동등하며, 관계대수로 표현한 식은 관계해석으로 표현할 수 있다.
- 질의어로 표현한다.

핵심 168 정규화의 개요

정규화란 함수적 종속성 등의 종속성 이론을 이용하여 잘못 설계된 관계형 스키마를 더 작은 속성의 세트로 쪼개어 바람직한 스키마로 만들어 가는 과정이다.

- 하나의 종속성이 하나의 릴레이션에 표현될 수 있도록 분해해가는 과정이라 할 수 있다.
- 정규형에는 제1정규형, 제2정규형, 제3정규형, BCNF형, 제4정규형, 제5정규형이 있으며, 차수가 높아질수록 만족시켜야 할 제약 조건이 늘어난다.
- 정규화는 데이터베이스의 논리적 설계 단계에서 수행한다.
- 정규화는 논리적 처리 및 품질에 큰 영향을 미친다.
- 정규화된 데이터 모델은 일관성, 정확성, 단순성, 비중복성, 안정성 등을 보장한다.

핵심 169 정규화의 목적

- 데이터 구조의 안정성 및 무결성을 유지한다.
- 어떠한 릴레이션이라도 데이터베이스 내에서 표현 가능하게 만든다.
- 효과적인 검색 알고리즘을 생성할 수 있다.
- 데이터 중복을 배제하여 이상(Anomaly)의 발생 방지 및 자료 저장 공간의 최소화가 가능하다.
- 데이터 삽입 시 릴레이션을 재구성할 필요성을 줄인다.
- 데이터 모형의 단순화가 가능하다.
- 속성의 배열 상태 검증이 가능하다.
- 개체와 속성의 누락 여부 확인이 가능하다.
- 자료 검색과 추출의 효율성을 추구한다.

핵심 170 이상(Anomaly)의 개념 및 종류

정규화를 거치지 않으면 데이터베이스 내에 데이터들이 불필요하게 중복되어 릴레이션 조작 시 예기치 못한 곤란한 현상이 발생하는데, 이를 이상(Anomaly)이라 하며 삽입 이상, 삭제 이상, 갱신 이상이 있다.

삽입 이상 (Insertion Anomaly)	릴레이션에 데이터를 삽입할 때 의도와는 상관없이 원하지 않은 값들도 함께 삽입되는 현상이다.
삭제 이상 (Deletion Anomaly)	릴레이션에서 한 튜플을 삭제할 때 의도와는 상관없는 값들도 함께 삭제되는 연쇄가 일어나는 현상이다.
갱신 이상 (Update Anomaly)	릴레이션에서 튜플에 있는 속성값을 갱신할 때 일부 튜플의 정보만 갱신되어 정보에 모순이 생기는 현상이다.

핵심 171 정규화 과정

비정규 릴레이션
↓ 도메인이 원자값
1NF
↓ 부분적 함수 종속 제거
2NF
↓ 이행적 함수 종속 제거
3NF
↓ 결정자이면서 후보키가 아닌 것 제거
BCNF
↓ 다치 종속 제거
4NF
↓ 조인 종속성 이용
5NF

정규화 단계 암기 요령

두부를 좋아하는 정규화가 두부가게에 가서 가게에 있는 두부를 다 달라고 말하니 주인이 깜짝 놀라며 말했다.

두부이걸다줘? ≒ 도부이결다조

도메인이 원자값
부분적 함수 종속 제거
이행적 함수 종속 제거
결정자이면서 후보키가 아닌 것 제거
다치 종속 제거
조인 종속성 이용

※ 이행적 종속(Transitive Dependency) 관계 : A → B이고 B → C일 때 A → C를 만족하는 관계를 의미

※ 함수적 종속(Functional Dependency)

- 함수적 종속은 데이터들이 어떤 기준값에 의해 종속되는 것을 의미한다.
- 예를 들어 〈수강〉 릴레이션이 (학번, 이름, 과목명)으로 되어 있을 때, '학번'이 결정되면 '과목명'에 상관없이 '학번'에는 항상 같은 '이름'이 대응된다. '학번'에 따라 '이름'이 결정될 때 '이름'을 '학번'에 함수종속적이라고 하며 '학번 → 이름'과 같이 쓴다.

핵심 172 반정규화의 개념

반정규화란 시스템의 성능 향상, 개발 및 운영의 편의성 등을 위해 정규화된 데이터 모델을 통합, 중복, 분리하는 과정으로, 의도적으로 정규화 원칙을 위배하는 행위이다.

- 반정규화를 수행하면 시스템의 성능이 향상되고 관리 효율성은 증가하지만 데이터의 일관성 및 정합성이 저하될 수 있다.
- 과도한 반정규화는 오히려 성능을 저하시킬 수 있다.
- 반정규화를 위해서는 사전에 데이터의 일관성과 무결성을 우선으로 할지, 데이터베이스의 성능과 단순화를 우선으로 할지를 결정해야 한다.
- 반정규화 방법에는 테이블 통합, 테이블 분할, 중복 테이블 추가, 중복 속성 추가 등이 있다.

핵심 173 반정규화 방법

테이블 통합	• 두 개의 테이블이 조인(Join)되는 경우가 많아 하나의 테이블로 합쳐 사용하는 것이 성능 향상에 도움이 될 경우 수행한다. • 두 개의 테이블에서 발생하는 프로세스가 동일하게 자주 처리되는 경우, 두 개의 테이블을 이용하여 항상 조회를 수행하는 경우 테이블 통합을 고려한다. • 테이블 통합의 종류에는 1:1 관계 테이블 통합, 1:N 관계 테이블 통합, 슈퍼타입/서브타입 테이블 통합이 있다.
테이블 분할	• 테이블을 수직 또는 수평으로 분할하는 것이다. • 수평 분할(Horizontal Partitioning) : 레코드(Record)를 기준으로 테이블을 분할하는 것이다. • 수직 분할(Vertical Partitioning) : 하나의 테이블에 속성이 너무 많을 경우 속성을 기준으로 테이블을 분할하는 것이다.
중복 테이블 추가	• 여러 테이블에서 데이터를 추출해서 사용해야 하거나 다른 서버에 저장된 테이블을 이용해야 하는 경우 중복 테이블을 추가하여 작업의 효율성을 향상시킬 수 있다. • 중복 테이블 추가 방법 : 집계 테이블의 추가, 진행 테이블의 추가, 특정 부분만을 포함하는 테이블의 추가
중복 속성 추가	조인해서 데이터를 처리할 때 데이터를 조회하는 경로를 단축하기 위해 자주 사용하는 속성을 하나 더 추가하는 것이다.

핵심 174 시스템 카탈로그

시스템 카탈로그는 시스템 그 자체에 관련이 있는 다양한 객체에 관한 정보를 포함하는 시스템 데이터베이스이다.

- 시스템 카탈로그 내의 각 테이블은 사용자를 포함하여 DBMS에서 지원하는 모든 데이터 객체에 대한 정의나 명세에 관한 정보를 유지 관리하는 시스템 테이블이다.
- 카탈로그들이 생성되면 데이터 사전(Data Dictionary)에 저장되기 때문에 좁은 의미로는 카탈로그를 데이터 사전이라고도 한다.
- 시스템 카탈로그에 저장된 정보를 메타 데이터(Meta-Data)라고 한다.
- 카탈로그 자체도 시스템 테이블로 구성되어 있어 일반 이용자도 SQL을 이용하여 내용을 검색해 볼 수 있다.
- INSERT, DELETE, UPDATE문으로 카탈로그를 갱신하는 것은 허용되지 않는다.
- 데이터베이스 시스템에 따라 상이한 구조를 갖는다.
- 카탈로그는 DBMS가 스스로 생성하고 유지한다.
- **카탈로그의 갱신** : 사용자가 SQL문을 실행시켜 기본 테이블, 뷰, 인덱스 등에 변화를 주면 시스템이 자동으로 갱신한다.

※ Data Directory

- 데이터 사전에 수록된 데이터를 실제로 접근하는 데 필요한 정보를 관리 유지하는 시스템이다.
- 시스템 카탈로그는 사용자와 시스템 모두 접근할 수 있지만 데이터 디렉터리는 시스템만 접근할 수 있다.

핵심 175 물리 데이터베이스 설계

물리 데이터베이스 설계는 논리적 구조로 표현된 논리적 데이터베이스를 디스크 등의 물리적 저장장치에 저장할 수 있는 물리적 구조의 데이터로 변환하는 과정이다.

- 물리적 데이터베이스 구조의 기본적인 데이터 단위는 저장 레코드(Stored Record)이다.
- 물리적 설계 단계에 꼭 포함되어야 할 것은 저장 레코드의 양식 설계, 레코드 집중(Record Clustering)의 분석 및 설계, 접근 경로 설계 등이다.
- 물리적 데이터베이스 구조는 여러 가지 타입의 저장 레코드 집합이라는 면에서 단순한 파일과 다르다.
- 물리적 데이터베이스 구조는 데이터베이스 시스템의 성능에 중대한 영향을 미친다.
- 물리적 설계 시 고려 사항
  - 인덱스 구조
  - 레코드 크기
  - 파일에 존재하는 레코드 개수
  - 파일에 대한 트랜잭션의 갱신과 참조 성향
  - 성능 향상을 위한 개념 스키마의 변경 여부 검토
  - 빈번한 질의와 트랜잭션들의 수행속도를 높이기 위한 고려
  - 시스템 운용 시 파일 크기의 변화 가능성

핵심 176 데이터베이스 저장 공간 설계

테이블(Table)

테이블은 데이터베이스의 가장 기본적인 객체로 로우(Row, 행)와 컬럼(Column, 열)으로 구성되어 있다.

- 테이블의 종류

일반 테이블	현재 사용되는 대부분의 DBMS에서 표준 테이블로 사용되는 테이블 형태이다.
클러스터드 인덱스 테이블(Clustered Index Table)	기본키(Primary Key)나 인덱스키의 순서에 따라 데이터가 저장되는 테이블이다.
파티셔닝 테이블(Partitioning Table)	대용량의 테이블을 작은 논리적 단위인 파티션(Partition)으로 나눈 테이블이다.
외부 테이블(External Table)	데이터베이스에서 일반 테이블처럼 이용할 수 있는 외부 파일로, 데이터베이스 내에 객체로 존재한다.
임시 테이블(Temporary Table)	트랜잭션이나 세션별로 데이터를 저장하고 처리할 수 있는 테이블이다.

컬럼(Column)

컬럼은 테이블의 열을 구성하는 요소로 데이터 타입(Data Type), 길이(Length) 등으로 정의된다.

- 데이터 타입은 데이터의 일관성 유지를 위해 사용되는 가장 기본적인 것으로, 도메인을 정의한 경우 도메인에 따라 데이터의 타입과 길이가 정의된다.
- 두 컬럼을 비교하는 연산에서 두 컬럼의 데이터 타입이나 길이가 다르면 DBMS 내부적으로 데이터 타입을 변환한 후 비교 연산을 수행한다.
- 참조 관계인 컬럼들은 데이터 타입과 길이가 일치해야 한다.

테이블스페이스(Tablespace)

테이블스페이스는 테이블이 저장되는 논리적인 영역으로, 하나의 테이블스페이스에 하나 또는 그 이상의 테이블을 저장할 수 있다.

- 테이블을 저장하면 논리적으로는 테이블스페이스에 저장되고, 물리적으로는 해당 테이블스페이스와 연관된 데이터 파일(Data File)에 저장된다.
- 데이터베이스를 테이블, 테이블스페이스, 데이터 파일로 나눠 관리하면 논리적 구성이 물리적 구성에 종속되지 않아 투명성이 보장된다.
- 테이블스페이스는 데이터베이스에 저장되는 내용에 따라 테이블, 인덱스, 임시(Temporary) 등의 용도로 구분하여 설계한다.

핵심 177 트랜잭션 분석 / CRUD 분석

트랜잭션(Transaction)

트랜잭션은 데이터베이스의 상태를 변환시키는 하나의 논리적 기능을 수행하기 위한 작업의 단위 또는 한꺼번에 모두 수행되어야 할 일련의 연산들을 의미한다.

- 트랜잭션은 데이터베이스 시스템에서 병행 제어 및 회복 작업 시 처리되는 작업의 논리적 단위로 사용된다.

• 트랜잭션의 특성

Atomicity (원자성)	• 트랜잭션의 연산은 데이터베이스에 모두 반영되도록 완료(Commit)되든지 아니면 전혀 반영되지 않도록 복구(Rollback)되어야 한다. • 트랜잭션 내의 모든 명령은 반드시 완벽히 수행되어야 하며, 모두가 완벽히 수행되지 않고 어느 하나라도 오류가 발생하면 트랜잭션 전부가 취소되어야 한다.
Consistency (일관성)	• 트랜잭션이 그 실행을 성공적으로 완료하면 언제나 일관성 있는 데이터베이스 상태로 변환한다. • 시스템이 가지고 있는 고정 요소는 트랜잭션 수행 전과 트랜잭션 수행 완료 후의 상태가 같아야 한다.
Isolation(독립성, 격리성, 순차성)	• 둘 이상의 트랜잭션이 동시에 병행 실행되는 경우 어느 하나의 트랜잭션 실행중에 다른 트랜잭션의 연산이 끼어들 수 없다. • 수행중인 트랜잭션은 완전히 완료될 때까지 다른 트랜잭션에서 수행 결과를 참조할 수 없다.
Durability (영속성, 지속성)	성공적으로 완료된 트랜잭션의 결과는 시스템이 고장나더라도 영구적으로 반영되어야 한다.

CRUD 분석

CRUD는 '생성(Create), 읽기(Read), 갱신(Update), 삭제(Delete)'의 앞 글자만 모아서 만든 용어이며, CRUD 분석은 데이터베이스 테이블에 변화를 주는 트랜잭션의 CRUD 연산에 대해 CRUD 매트릭스를 작성하여 분석하는 것이다.

- CRUD 분석으로 테이블에 발생되는 트랜잭션의 주기별 발생 횟수를 파악하고 연관된 테이블들을 분석하면 테이블에 저장되는 데이터의 양을 유추할 수 있다.

CRUD 매트릭스

CRUD 매트릭스는 2차원 형태의 표로서, 행(Row)에는 프로세스를, 열(Column)에는 테이블을, 행과 열이 만나는 위치에는 프로세스가 테이블에 발생시키는 변화를 표시하는 업무 프로세스와 데이터 간 상관 분석표이다.

- CRUD 매트릭스를 통해 프로세스의 트랜잭션이 테이블에 수행하는 작업을 검증한다.

트랜잭션 분석

트랜잭션 분석의 목적은 CRUD 매트릭스를 기반으로 테이블에 발생하는 트랜잭션 양을 분석하여 테이블에 저장되는 데이터의 양을 유추하고 이를 근거로 DB 용량을 산정하고 DB 구조를 최적화하는 것이다.

- 트랜잭션 분석은 업무 개발 담당자가 수행한다.

트랜잭션 분석서

트랜잭션 분석서는 단위 프로세스와 CRUD 매트릭스를 이용하여 작성하며, 구성 요소에는 단위 프로세스, CRUD 연산, 테이블명, 컬럼명, 테이블 참조 횟수, 트랜잭션 수, 발생 주기 등이 있다.

핵심 178 인덱스(Index)

인덱스는 데이터 레코드를 빠르게 접근하기 위해 〈키 값, 포인터〉 쌍으로 구성되는 데이터 구조이다.

- 인덱스는 데이터가 저장된 물리적 구조와 밀접한 관계가 있다.
- 인덱스는 레코드가 저장된 물리적 구조에 접근하는 방법을 제공한다.
- 인덱스를 통해서 파일의 레코드에 대한 액세스를 빠르게 수행할 수 있다.
- 레코드의 삽입과 삭제가 수시로 일어나는 경우에는 인덱스의 개수를 최소로 하는 것이 효율적이다.
- 인덱스의 종류

트리 기반 인덱스	인덱스를 저장하는 블록들이 트리 구조를 이루고 있는 것으로, 상용 DBMS에서는 트리 구조 기반의 B+ 트리 인덱스를 주로 활용한다.
비트맵 인덱스	인덱스 컬럼의 데이터를 Bit 값인 0 또는 1로 변환하여 인덱스 키로 사용하는 방법이다.
함수 기반 인덱스	• 컬럼의 값 대신 컬럼에 특정 함수(Function)나 수식(Expression)을 적용하여 산출된 값을 사용하는 것으로, B+ 트리 인덱스 또는 비트맵 인덱스를 생성하여 사용한다. • 데이터를 입력하거나 수정할 때 함수를 적용해야 하므로 부하가 발생할 수 있다.
비트맵 조인 인덱스	다수의 조인된 객체로 구성된 인덱스로, 단일 객체로 구성된 일반적인 인덱스와 액세스 방법이 다르다.
도메인 인덱스	개발자가 필요한 인덱스를 직접 만들어 사용하는 것으로, 확장형 인덱스(Extensible Index)라고도 한다.

※ 클러스터드 인덱스 / 넌클러스터드 인덱스

- 클러스터드 인덱스(Clustered Index) : 인덱스 키의 순서에 따라 데이터가 정렬되어 저장되는 방식
- 넌클러스터드 인덱스(Non-Clustered Index) : 인덱스의 키 값만 정렬되어 있을 뿐 실제 데이터는 정렬되지 않는 방식

핵심 179 뷰(View)

뷰는 사용자에게 접근이 허용된 자료만을 제한적으로 보여주기 위해 하나 이상의 기본 테이블로부터 유도된, 이름을 가지는 가상 테이블이다.

- 뷰는 저장장치 내에 물리적으로 존재하지 않지만, 사용자에게는 있는 것처럼 간주된다.
- 뷰는 데이터 보정 작업, 처리 과정 시험 등 임시적인 작업을 위한 용도로 활용된다.

뷰(View)의 특징

- 뷰는 기본 테이블로부터 유도된 테이블이기 때문에 기본 테이블과 같은 형태의 구조를 사용하며, 조작도 기본 테이블과 거의 같다.
- 뷰는 가상 테이블이기 때문에 물리적으로 구현되어 있지 않다.
- 데이터의 논리적 독립성을 제공할 수 있다.
- 필요한 데이터만 뷰로 정의해서 처리할 수 있기 때문에 관리가 용이하고 명령문이 간단해진다.
- 뷰를 통해서만 데이터에 접근하게 하면 뷰에 나타나지 않는 데이터를 안전하게 보호하는 효율적인 기법으로 사용할 수 있다.
- 기본 테이블의 기본키를 포함한 속성(열) 집합으로 뷰를 구성해야만 삽입, 삭제, 갱신 연산이 가능하다.
- 일단 정의된 뷰는 다른 뷰의 정의에 기초가 될 수 있다.
- 뷰를 정의할 때는 CREATE문, 제거할 때는 DROP문을 사용한다.

뷰(View)의 장 · 단점

장점	단점
• 논리적 데이터 독립성을 제공한다. • 동일 데이터에 대해 동시에 여러 사용자의 상이한 응용이나 요구를 지원해 준다. • 사용자의 데이터 관리를 간단하게 해준다. • 접근 제어를 통한 자동 보안이 제공된다.	• 독립적인 인덱스를 가질 수 없다. • 뷰의 정의를 변경할 수 없다. • 뷰로 구성된 내용에 대한 삽입, 삭제, 갱신 연산에 제약이 따른다.

핵심 180 클러스터(Cluster)

클러스터는 데이터 저장 시 데이터 액세스 효율을 향상시키기 위해 동일한 성격의 데이터를 동일한 데이터 블록에 저장하는 물리적 저장 방법이다.

- 클러스터링키로 지정된 컬럼 값의 순서대로 저장되고, 여러 개의 테이블이 하나의 클러스터에 저장된다.

클러스터(Cluster)의 특징

- 클러스터링 된 테이블은 데이터 조회 속도는 향상시키지만 데이터 입력, 수정, 삭제에 대한 성능은 저하시킨다.
- 클러스터는 데이터의 분포도가 넓을수록 유리하다.
- 데이터 분포도가 넓은 테이블을 클러스터링 하면 저장 공간을 절약할 수 있다.
- 클러스터링된 테이블은 클러스터링키 열을 공유하므로 저장 공간이 줄어든다.
- 대용량을 처리하는 트랜잭션은 전체 테이블을 스캔하는 일이 자주 발생하므로 클러스터링을 하지 않는 것이 좋다.
- 파티셔닝된 테이블에는 클러스터링을 할 수 없다.

핵심 181 파티션(Patition)

데이터베이스에서 파티션은 대용량의 테이블이나 인덱스를 작은 논리적 단위인 파티션으로 나누는 것을 말한다.

- 대용량 DB의 경우 중요한 몇 개의 테이블에만 집중되어 데이터가 증가되므로, 이런 테이블들을 작은 단위로 나눠 분산시키면 성능 저하를 방지할 뿐만 아니라 데이터 관리도 쉬워진다.

파티션의 장 · 단점

장점	• 데이터 접근 시 액세스 범위를 줄여 쿼리 성능이 향상된다. • 파티션별로 데이터가 분산되어 저장되므로 디스크의 성능이 향상된다. • 파티션별로 백업 및 복구를 수행하므로 속도가 빠르다. • 시스템 장애 시 데이터 손상 정도를 최소화할 수 있다. • 데이터 가용성이 향상된다. • 파티션 단위로 입 · 출력을 분산시킬 수 있다.
단점	• 하나의 테이블을 세분화하여 관리하므로 세심한 관리가 요구된다. • 테이블간 조인에 대한 비용이 증가한다. • 용량이 작은 테이블에 파티셔닝을 수행하면 오히려 성능이 저하된다.

파티션의 종류

범위 분할 (Range Partitioning)	지정한 열의 값을 기준으로 분할한다. 예 일별, 월별, 분기별 등
해시 분할 (Hash Partitioning)	• 해시 함수를 적용한 결과 값에 따라 데이터를 분할한다. • 특정 파티션에 데이터가 집중되는 범위 분할의 단점을 보완한 것으로, 데이터를 고르게 분산할 때 유용하다. • 특정 데이터가 어디에 있는지 판단할 수 없다. • 고객번호, 주민번호 등과 같이 데이터가 고른 컬럼에 효과적이다.
조합 분할 (Composite Partitioning)	• 범위 분할로 분할한 다음 해시 함수를 적용하여 다시 분할하는 방식이다. • 범위 분할한 파티션이 너무 커서 관리가 어려울 때 유용하다.

핵심 182 데이터베이스 용량 설계

데이터베이스 용량 설계

데이터베이스 용량 설계는 데이터가 저장될 공간을 정의하는 것이다.

- 데이터베이스 용량을 설계할 때는 테이블에 저장할 데이터양과 인덱스, 클러스터 등이 차지하는 공간 등을 예측하여 반영해야 한다.

데이터베이스 용량 설계의 목적

- 데이터베이스의 용량을 정확히 산정하여 디스크의 저장 공간을 효과적으로 사용하고 확장성 및 가용성을 높인다.
- 디스크의 특성을 고려하여 설계함으로써 디스크의 입 · 출력 부하를 분산시키고 채널의 병목 현상을 최소화한다.
- 디스크에 대한 입 · 출력 경합이 최소화되도록 설계함으로써 데이터 접근성이 향상된다.
- 데이터 접근성을 향상시키는 설계 방법
  - 테이블의 테이블스페이스와 인덱스의 테이블스페이스를 분리하여 구성한다.
  - 테이블스페이스와 임시 테이블스페이스를 분리하여 구성한다.
  - 테이블을 마스터 테이블과 트랜잭션 테이블로 분류한다.
- 데이터베이스에 생성되는 오브젝트의 익스텐트 발생을 최소화하여 성능을 향상시킨다.
- 데이터베이스 용량을 정확히 분석하여 테이블과 인덱스에 적합한 저장 옵션을 지정한다.

핵심 183 분산 데이터베이스 설계

분산 데이터베이스는 논리적으로는 하나의 시스템에 속하지만 물리적으로는 네트워크를 통해 연결된 여러 개의 컴퓨터 사이트(Site)에 분산되어 있는 데이터베이스를 말한다.

분산 데이터베이스의 목표

- 위치 투명성(Location Transparency) : 액세스하려는 데이터베이스의 실제 위치를 알 필요 없이 단지 데이터베이스의 논리적인 명칭만으로 액세스할 수 있다.
- 중복 투명성(Replication Transparency) : 동일 데이터가 여러 곳에 중복되어 있더라도 사용자는 마치 하나의 데이터만 존재하는 것처럼 사용하고, 시스템은 자동으로 여러 자료에 대한 작업을 수행한다.
- 병행 투명성(Concurrency Transparency) : 분산 데이터베이스와 관련된 다수의 트랜잭션들이 동시에 실현되더라도 그 트랜잭션의 결과는 영향을 받지 않는다.

분산 데이터베이스의 장 · 단점

장점	단점
• 지역 자치성이 높다. • 자료의 공유성이 향상된다. • 분산 제어가 가능하다. • 시스템 성능이 향상된다. • 중앙 컴퓨터의 장애가 전체 시스템에 영향을 끼치지 않는다. • 효용성과 융통성이 높다. • 신뢰성 및 가용성이 높다. • 점진적 시스템 용량 확장이 용이하다.	• DBMS가 수행할 기능이 복잡하다. • 데이터베이스 설계가 어렵다. • 소프트웨어 개발 비용이 증가한다. • 처리 비용이 증가한다. • 잠재적 오류가 증가한다.

분산 데이터베이스 설계

분산 데이터베이스 설계는 애플리케이션이나 사용자가 분산되어 저장된 데이터에 접근하게 하는 것을 목적으로 한다.

- 분산 설계 방법에는 테이블 위치 분산, 분할(Fragmentation), 할당(Allocation)이 있다.
  - 테이블 위치 분산 : 데이터베이스의 테이블을 각기 다른 서버에 분산시켜 배치하는 방법
  - 분할 : 테이블의 데이터를 분할하여 분산시키는 것

▶ 주요 분할 방법

수평 분할	특정 속성의 값을 기준으로 행(Row) 단위로 분할
수직 분할	데이터 컬럼(속성) 단위로 분할

  - 할당 : 동일한 분할을 여러 개의 서버에 생성하는 분산 방법으로, 중복이 없는 할당(Allocation)과 중복이 있는 할당(Allocation)으로 나뉨

핵심 184 데이터베이스 이중화 / 서버 클러스터링

데이터베이스 이중화(Database Replication)

데이터베이스 이중화는 시스템 오류로 인한 데이터베이스 서비스 중단이나 물리적 손상 발생 시 이를 복구하기 위해 동일한 데이터베이스를 복제하여 관리하는 것이다.

- 데이터베이스 이중화를 수행하면 하나 이상의 데이터베이스가 항상 같은 상태를 유지하므로 데이터베이스에 문제가 발생하면 복제된 데이터베이스를 이용하여 즉시 문제를 해결할 수 있다.
- 데이터베이스 이중화의 분류

Eager 기법	트랜잭션 수행 중 데이터 변경이 발생하면 이중화된 모든 데이터베이스에 즉시 전달하여 변경 내용이 즉시 적용되도록 하는 기법
Lazy 기법	트랜잭션의 수행이 종료되면 변경 사실을 새로운 트랜잭션에 작성하여 각 데이터베이스에 전달되는 기법으로, 데이터베이스마다 새로운 트랜잭션이 수행되는 것으로 간주된다.

- 데이터베이스 이중화 구성 방법

활동-대기 (Active-Standby) 방법	• 한 DB가 활성 상태로 서비스하고 있으면 다른 DB는 대기하고 있다가 활성 DB에 장애가 발생하면 대기 상태에 있던 DB가 자동으로 모든 서비스를 대신 수행한다. • 구성 방법과 관리가 쉬워 많은 기업에서 이용된다.
활동-활동 (Active-Active) 방법	• 두 개의 DB가 서로 다른 서비스를 제공하다가 둘 중 한쪽 DB에 문제가 발생하면 나머지 다른 DB가 서비스를 제공한다. • 두 DB가 모두 처리를 하기 때문에 처리율이 높지만 구성 방법 및 설정이 복잡하다.

클러스터링(Clustering)

클러스터링은 두 대 이상의 서버를 하나의 서버처럼 운영하는 기술이다.

- 클러스터링은 서버 이중화 및 공유 스토리지를 사용하여 서버의 고가용성을 제공한다.
- 클러스터링에는 고가용성 클러스터링과 병렬 처리 클러스터링이 있다.
  - 고가용성 클러스터링 : 하나의 서버에 장애가 발생하면 다른 노드(서버)가 받아 처리하여 서비스 중단을 방지하는 방식으로, 일반적으로 언급되는 클러스터링이 고가용성 클러스터링이다.
  - 병렬 처리 클러스터링 : 전체 처리율을 높이기 위해 하나의 작업을 여러 개의 서버에서 분산하여 처리하는 방식이다.

핵심 185 암호화(Encryption)

암호화는 데이터를 보낼 때 송신자가 지정한 수신자 이외에는 그 내용을 알 수 없도록 평문을 암호문으로 변환하는 것이다.

- 암호화(Encryption) 과정 : 암호화되지 않은 평문을 정보 보호를 위해 암호문으로 바꾸는 과정
- 복호화(Decryption) 과정 : 암호문을 원래의 평문으로 바꾸는 과정
- 암호화 기법에는 개인키 암호 방식과 공개키 암호 방식이 있다.

개인키 암호 방식(Private Key Encryption) = 비밀키 암호 방식

비밀키 암호화 기법은 동일한 키로 데이터를 암호화하고 복호화한다.

- 비밀키 암호화 기법은 대칭 암호 방식 또는 단일키 암호화 기법이라고도 한다.
- 비밀키는 제3자에게는 노출시키지 않고 데이터베이스 사용 권한이 있는 사용자만 나누어 가진다.
- **종류** : 전위 기법, 대체 기법, 대수 기법, 합성 기법 (DES, LUCIFER)

공개키 암호 방식(Public Key Encryption)

공개키 암호화 기법은 서로 다른 키로 데이터를 암호화하고 복호화한다.

- 데이터를 암호화할 때 사용하는 키(공개키, Public Key)는 데이터베이스 사용자에게 공개하고, 복호화할 때의 키(비밀키, Secret Key)는 관리자가 비밀리에 관리 하는 방법이다.
- 공개키 암호화 기법은 비대칭 암호 방식이라고도 하며, 대표적으로 RSA(Rivest Shamir Adleman)가 있다.

핵심 186 데이터베이스 보안 – 접근통제

접근통제

접근통제는 데이터가 저장된 객체와 이를 사용하려는 주체 사이의 정보 흐름을 제한하는 것이다.

- 접근통제 기술

임의 접근통제 (DAC, Discretionary Access Control)	• 임의 접근통제는 데이터에 접근하는 사용자의 신원에 따라 접근 권한을 부여하는 방식이다. • 데이터 소유자가 접근통제 권한을 지정하고 제어한다. • 객체를 생성한 사용자가 생성된 객체에 대한 모든 권한을 부여받고, 부여된 권한을 다른 사용자에게 허가할 수도 있다. • 임의 접근통제에 사용되는 SQL 명령어에는 GRANT와 REVOKE가 있다.
강제 접근통제 (MAC, Mandatory Access Control)	• 강제 접근통제는 주체와 객체의 등급을 비교하여 접근 권한을 부여하는 방식이다. • 시스템이 접근통제 권한을 지정한다. • 데이터베이스 객체별로 보안 등급을 부여할 수 있고, 사용자별로 인가 등급을 부여할 수 있다. • 주체는 자신보다 보안 등급이 높은 객체에 대해 읽기, 수정, 등록이 모두 불가능하고, 보안 등급이 같은 객체에 대해서는 읽기, 수정, 등록이 가능하고, 보안 등급이 낮은 객체는 읽기가 가능하다.
역할기반 접근통제 (RBAC, Role Based Access Control)	• 역할기반 접근통제는 사용자의 역할에 따라 접근 권한을 부여하는 방식이다. • 중앙관리자가 접근 통제 권한을 지정한다. • 임의 접근통제와 강제 접근통제의 단점을 보완하였으며, 다중 프로그래밍 환경에 최적화된 방식이다. • 중앙관리자가 역할마다 권한을 부여하면, 책임과 자질에 따라 역할을 할당받은 사용자들은 역할에 해당하는 권한을 사용할 수 있다.

- **접근통제의 3요소** : 접근통제 정책, 접근통제 메커니즘, 접근통제 보안모델

접근통제 정책

접근통제 정책은 어떤 주체가(Who)가 언제(When), 어디서(Where), 어떤 객체(What)에게, 어떤 행위(How)에 대한 허용 여부를 정의하는 것으로, 신분 기반 정책, 규칙 기반 정책, 역할 기반 정책이 있다.

신분 기반 정책	• 주체나 그룹의 신분에 근거하여 객체의 접근을 제한하는 방법으로, IBP와 GBP가 있다. • IBP(Individual-Based Policy) : 최소 권한 정책으로, 단일 주체에게 하나의 객체에 대한 허가를 부여한다. • GBP(Group-Based Policy) : 복수 주체에 하나의 객체에 대한 허가를 부여한다.
규칙 기반 정책	• 주체가 갖는 권한에 근거하여 객체의 접근을 제한하는 방법으로, MLP와 CBP가 있다. • MLP(Multi-Level Policy) : 사용자 및 객체별로 지정된 기밀 분류에 따른 정책 • CBP(Compartment-Based Policy) : 집단별로 지정된 기밀 허가에 따른 정책
역할 기반 정책	GBP의 변형된 정책으로, 주체의 신분이 아니라 주체가 맡은 역할에 근거하여 객체의 접근을 제한하는 방법이다. 예 인사 담당자, DBA 등

접근통제 매커니즘

접근통제 매커니즘은 정의된 접근통제 정책을 구현하는 기술적인 방법으로, 접근통제 목록, 능력 리스트, 보안 등급, 패스워드, 암호화 등이 있다.

접근통제 보안 모델

접근통제 보안 모델은 보안 정책을 구현하기 위한 정형화된 모델로, 기밀성 모델, 무결성 모델, 접근통제 모델이 있다.

기밀성 모델	• 군사적인 목적으로 개발된 최초의 수학적 모델로, 기밀성 보장이 최우선인 모델이다. • 기밀성 모델은 군대 시스템 등 특수 환경에서 주로 사용된다.

무결성 모델	• 기밀성 모델에서 발생하는 불법적인 정보 변경을 방지하기 위해 무결성을 기반으로 개발된 모델이다. • 무결성 모델은 데이터의 일관성 유지에 중점을 두어 개발되었다.
접근통제 모델	접근통제 메커니즘을 보안 모델로 발전시킨 것으로, 대표적으로 접근통제 행렬(Access Control Matrix)이 있다.

핵심 187 데이터베이스 백업

데이터베이스 백업

데이터베이스 백업은 전산 장비의 장애에 대비하여 데이터베이스에 저장된 데이터를 보호하고 복구하기 위한 작업으로, 치명적인 데이터 손실을 막기 위해서는 데이터베이스를 정기적으로 백업해야 한다.

- 데이터베이스 관리 시스템(DBMS)은 데이터베이스 파괴 및 실행 중단이 발생하면 이를 복구할 수 있는 기능을 제공한다.

로그 파일

로그 파일은 데이터베이스의 처리 내용이나 이용 상황 등 상태 변화를 시간의 흐름에 따라 모두 기록한 파일로, 데이터베이스의 복구를 위해 필요한 가장 기본적인 자료이다.

- 로그 파일을 기반으로 데이터베이스를 과거 상태로 복귀(UNDO)시키거나 현재 상태로 재생(REDO)시켜 데이터베이스 상태를 일관성 있게 유지할 수 있다.
- 로그 파일은 트랜잭션 시작 시점, Rollback 시점, 데이터 입력, 수정 삭제 시점 등에서 기록된다.

데이터베이스 복구 알고리즘

NO-UNDO/ REDO	• 데이터베이스 버퍼의 내용을 비동기적으로 갱신한 경우의 복구 알고리즘 • NO-UNDO : 트랜잭션 완료 전에는 변경 내용이 데이터베이스에 기록되지 않으므로 취소할 필요가 없다. • REDO : 트랜잭션 완료 후 데이터베이스 버퍼에는 기록되어 있고, 저장매체에는 기록되지 않았으므로 트랜잭션 내용을 다시 실행해야 한다.
UNDO/ NO-REDO	• 데이터베이스 버퍼의 내용을 동기적으로 갱신한 경우의 복구 알고리즘 • UNDO : 트랜잭션 완료 전에 시스템이 파손되었다면 변경된 내용을 취소한다. • NO-REDO : 트랜잭션 완료 전에 데이터베이스 버퍼 내용을 이미 저장 매체에 기록했으므로 트랜잭션 내용을 다시 실행할 필요가 없다.
UNDO/ REDO	• 데이터베이스 버퍼의 내용을 동기/비동기적으로 갱신한 경우의 복구 알고리즘 • 데이터베이스 기록 전에 트랜잭션이 완료될 수 있으므로 완료된 트랜잭션이 데이터베이스에 기록되지 못했다면 다시 실행해야 한다.
NO-UNDO/ NO-REDO	• 데이터베이스 버퍼의 내용을 동기적으로 저장 매체에 기록하지만 데이터베이스와는 다른 영역에 기록한 경우의 복구 알고리즘 • NO-UNDO : 변경 내용은 데이터베이스와 다른 영역에 기록되어 있으므로 취소할 필요가 없다. • NO-REDO : 다른 영역에 이미 기록되어 있으므로 트랜잭션을 다시 실행할 필요가 없다.

백업 종류

백업 종류는 복구 수준에 따라서 운영체제를 이용하는 물리 백업과 DBMS 유틸리티를 이용하는 논리 백업으로 나뉜다.

- **물리 백업** : 데이터베이스 파일을 백업하는 방법으로, 백업 속도가 빠르고 작업이 단순하지만 문제 발생 시 원인 파악 및 문제 해결이 어렵다.
- **논리 백업** : DB 내의 논리적 객체들을 백업하는 방법으로, 복원 시 데이터 손상을 막고 문제 발생 시 원인 파악 및 해결이 수월하지만 백업/복원 시 시간이 많이 소요된다.

구분	설명	복구 수준
물리 백업	로그 파일 백업 실시	완전 복구
	로그 파일 백업 없음	백업 시점까지 복구
논리 백업	DBMS 유틸리티	

핵심 188 스토리지

스토리지(Storage)의 개요

스토리지는 단일 디스크로 처리할 수 없는 대용량의 데이터를 저장하기 위해 서버와 저장장치를 연결하는 기술이다.

DAS(Direct Attached Storage)

DAS는 서버와 저장장치를 전용 케이블로 직접 연결하는 방식으로, 일반 가정에서 컴퓨터에 외장하드를 연결하는 것이 여기에 해당된다.

- 서버에서 저장장치를 관리한다.
- 저장장치를 직접 연결하므로 속도가 빠르고 설치 및 운영이 쉽다.

- 초기 구축 비용 및 유지보수 비용이 저렴하다.
- 확장성 및 유연성이 상대적으로 떨어진다.

NAS(Network Attached Storage)

- NAS는 서버와 저장장치를 네트워크를 통해 연결하는 방식이다.
- 별도의 파일 관리 기능이 있는 NAS Storage가 내장된 저장장치를 직접 관리한다.
- Ethernet 스위치를 통해 다른 서버에서도 스토리지에 접근할 수 있어 파일 공유가 가능하고, 장소에 구애받지 않고 저장장치에 쉽게 접근할 수 있다.
- DAS에 비해 확장성 및 유연성이 우수하다.
- 접속 증가 시 성능이 저하될 수 있다.

SAN(Storage Area Network)

SAN은 DAS의 빠른 처리와 NAS의 파일 공유 장점을 혼합한 방식으로, 서버와 저장장치를 연결하는 전용 네트워크를 별도로 구성하는 방식이다.

- 파이버 채널 스위치는 서버나 저장장치를 광케이블로 연결하므로 처리 속도가 빠르다.
- 서버들이 저장장치 및 파일을 공유할 수 있다.
- 확장성, 유연성, 가용성이 뛰어나다.

핵심 189 논리 데이터 모델의 물리 데이터 모델 변환

엔티티(Entity)를 테이블로 변환

논리 데이터 모델에서 정의된 엔티티를 물리 데이터 모델의 테이블로 변환하는 것이다.

- 변환 규칙

논리적 설계(데이터 모델링)	물리적 설계
엔티티(Entity)	테이블(Table)
속성(Attribute)	컬럼(Column)
주 식별자(Primary Identifier)	기본키(Primary Key)
외부 식별자(Foreign Identifier)	외래키(Foreign Key)
관계(Relationship)	관계(Relationship)

슈퍼타입/서브타입을 테이블로 변환

슈퍼타입/서브타입은 논리 데이터 모델에서 이용되는 형태이므로 물리 데이터 모델을 설계할 때는 슈퍼타입/서브타입을 테이블로 변환해야 한다.

- **슈퍼타입 기준 테이블 변환** : 서브타입을 슈퍼타입에 통합하여 하나의 테이블로 만드는 것
  - 장 · 단점

장점	• 데이터의 액세스가 상대적으로 용이하다. • 뷰를 이용하여 각각의 서브타입만을 액세스하거나 수정할 수 있다. • 서브타입 구분이 없는 임의 집합에 대한 처리가 용이하다. • 여러 테이블을 조인하지 않아도 되므로 수행 속도가 빨라진다. • SQL 문장 구성이 단순해진다.
단점	• 테이블의 컬럼이 증가하므로 디스크 저장 공간이 증가한다. • 처리마다 서브타입에 대한 구분(TYPE)이 필요한 경우가 많이 발생한다. • 인덱스 크기의 증가로 인덱스의 효율이 떨어진다.

- **서브타입 기준 테이블 변환** : 슈퍼타입 속성들을 각각의 서브타입에 추가하여 서브타입들을 개별적인 테이블로 만드는 것
  - 장 · 단점

장점	• 각 서브타입 속성들의 선택 사양이 명확한 경우에 유리하다. • 처리할 때마다 서브타입 유형을 구분할 필요가 없다. • 여러 개의 테이블로 통합하므로 테이블당 크기가 감소하여 전체 테이블 스캔시 유리하다.
단점	• 수행 속도가 감소할 수 있다. • 복잡한 처리를 하는 SQL의 통합이 어렵다. • 부분 범위에 대한 처리가 곤란해진다. • 여러 테이블을 통합한 뷰는 조회만 가능하다. • UID(Unique Identifier, 식별자)의 유지 관리가 어렵다.

- **개별타입 기준 테이블 변환** : 슈퍼타입과 서브타입들을 각각의 개별적인 테이블로 변환하는 것이다.
  - 슈퍼타입과 서브타입 테이블들 사이에는 각각 1:1 관계가 형성된다.
  - 장 · 단점

장점	• 저장 공간이 상대적으로 작다. • 슈퍼타입 또는 서브타입 각각의 테이블에 속한 정보만 조회하는 경우 문장 작성이 용이하다.
단점	슈퍼타입 또는 서브타입의 정보를 같이 처리하면 항상 조인이 발생하여 성능이 저하된다.

속성을 컬럼으로 변환

논리 데이터 모델에서 정의한 속성을 물리 데이터 모델의 컬럼으로 변환한다.

- Primary UID를 기본키로 변환 : 논리 데이터 모델에서의 Primary UID는 물리 데이터 모델의 기본키로 만든다.
- Primary UID(관계의 UID Bar)를 기본키로 변환 : 다른 엔티티와의 관계로 인해 생성된 Primary UID는 물리 데이터 모델의 기본키로 만든다.
- Secondary(Alternate) UID를 유니크키로 변환 : 논리 모델링에서 정의된 Secondary UID 및 Alternate Key는 물리 모델에서 유니크키로 만든다.

관계를 외래키로 변환

논리 데이터 모델에서 정의된 관계는 기본키와 이를 참조하는 외래키로 변환한다.

- 다음은 개체 A, B로 이루어진 E-R 모델을 관계 테이블로 변환하는 방법이다.
  - 1:1 관계 : 개체 A의 기본키를 개체 B의 외래키로 추가하거나 개체 B의 기본키를 개체 A의 외래키로 추가하여 표현한다.
  - 1:M 관계 : 개체 A의 기본키를 개체 B의 외래키로 추가하여 표현하거나 별도의 테이블로 표현한다.
  - N:M 관계 : 릴레이션 A와 B의 기본키를 모두 포함한 별도의 릴레이션으로 표현한다. 이때 생성된 별도의 릴레이션을 교차 릴레이션(Intersection Relation) 또는 교차 엔티티(Intersection Entity)라고 한다.
  - 1:M 순환 관계 : 개체 A에 개체 A의 기본키를 참조하는 외래키 컬럼을 추가하여 표현한다. 데이터의 계층 구조를 표현하기 위해 주로 사용된다.

핵심 190 물리 데이터 모델 품질 검토

물리 데이터 모델 품질 검토

- 물리 데이터 모델의 품질 검토는 물리 데이터 모델을 설계하고 데이터베이스 객체를 생성한 후 개발 단계로 넘어가기 전에 모델러와 이해관계자들이 모여 수행한다.
- 물리 데이터 모델은 시스템 성능에 직접적인 영향을 미치므로 향후 발생할 문제에 대해 면밀히 검토해야 한다.
- 물리 데이터 모델 품질 검토의 목적은 데이터베이스의 성능 향상과 오류 예방이다.
- 물리 데이터 모델을 검토하려면 모든 이해관계자가 동의하는 검토 기준이 필요하다.

물리 데이터 모델 품질 기준

정확성	데이터 모델이 요구사항이나 업무 규칙, 표기법에 따라 정확하게 표현되었음을 의미한다.
완전성	데이터 모델이 데이터 모델의 구성 요소를 누락 없이 정의하고 요구사항이나 업무 영역을 누락 없이 반영하였음을 의미한다.
준거성	데이터 모델이 데이터 표준, 표준화 규칙, 법적 요건 등을 정확하게 준수하였음을 의미한다.
최신성	데이터 모델이 최근의 이슈나 현행 시스템을 반영하고 있음을 의미한다.
일관성	데이터 모델이 표현상의 일관성을 유지하고 있음을 의미한다.
활용성	작성된 모델과 설명을 사용자가 충분히 이해할 수 있고, 업무 변화에 따른 데이터 구조의 변경이 최소화될 수 있도록 설계되었음을 의미한다.

- 물리 데이터 모델의 품질 기준은 조직 혹은 업무 상황에 따라 가감하거나 변형하여 사용한다.

핵심 191 DDL(Data Define Language, 데이터 정의어)

- DDL은 SCHEMA, DOMAIN, TABLE, VIEW, INDEX를 정의하거나 변경 또는 삭제할 때 사용하는 언어이다.
- 논리적 데이터 구조와 물리적 데이터 구조의 사상을 정의한다.
- 데이터베이스 관리자나 데이터베이스 설계자가 사용한다.
- DDL(데이터 정의어)의 세 가지 유형

명령어	기능
CREATE	SCHEMA, DOMAIN, TABLE, VIEW, INDEX를 정의한다.
ALTER	TABLE에 대한 정의를 변경하는 데 사용한다.
DROP	SCHEMA, DOMAIN, TABLE, VIEW, INDEX를 삭제한다.

핵심 192 DML(Data Manipulation Language, 데이터 조작어)

- DML은 데이터베이스 사용자가 응용 프로그램이나 질의어를 통하여 저장된 데이터를 실질적으로 처리하는 데 사용되는 언어이다.
- 데이터베이스 사용자와 데이터베이스 관리 시스템 간의 인터페이스를 제공한다.
- DML(데이터 조작어)의 네 가지 유형

명령어	기능
SELECT	테이블에서 조건에 맞는 튜플을 검색한다.
INSERT	테이블에 새로운 튜플을 삽입한다.
DELETE	테이블에서 조건에 맞는 튜플을 삭제한다.
UPDATE	테이블에서 조건에 맞는 튜플의 내용을 변경한다.

핵심 193 DCL(Data Control Language, 데이터 제어어)

- DCL은 데이터의 보안, 무결성, 회복, 병행 수행 제어 등을 정의하는 데 사용되는 언어이다.
- 데이터베이스 관리자가 데이터 관리를 목적으로 사용한다.
- DCL(데이터 제어어)의 종류

명령어	기능
COMMIT	명령에 의해 수행된 결과를 실제 물리적 디스크로 저장하고, 데이터베이스 조작 작업이 정상적으로 완료되었음을 관리자에게 알려준다.
ROLLBACK	데이터베이스 조작 작업이 비정상적으로 종료되었을 때 원래의 상태로 복구한다.
GRANT	데이터베이스 사용자에게 사용 권한을 부여한다.
REVOKE	데이터베이스 사용자의 사용 권한을 취소한다.

핵심 194 CREATE TABLE

CREATE TABLE은 테이블을 정의하는 명령문이다.

표기 형식

```
CREATE TABLE 테이블명
        (속성명 데이터_타입 [DEFAULT 기본값] [NOT NULL], …
        [, PRIMARY KEY(기본키_속성명, …)]
        [, UNIQUE(대체키_속성명, …)]
        [, FOREIGN KEY(외래키_속성명, …)]
                [REFERENCES 참조테이블(기본키_속성명, …)]
                [ON DELETE 옵션]
                [ON UPDATE 옵션]
        [, CONSTRAINT 제약조건명] [CHECK (조건식)]);
```

- 기본 테이블에 포함될 모든 속성에 대하여 속성명과 그 속성의 데이터 타입, 기본값, NOT NULL 여부를 지정한다.
- PRIMARY KEY : 기본키로 사용할 속성 또는 속성의 집합을 지정한다.
- UNIQUE : 대체키로 사용할 속성 또는 속성의 집합을 지정하는 것으로 UNIQUE로 지정한 속성은 중복된 값을 가질 수 없다.
- FOREIGN KEY ~ REFERENCES ~
  - 참조할 다른 테이블과 그 테이블을 참조할 때 사용할 외래키 속성을 지정한다.
  - 외래키가 지정되면 참조 무결성의 CASCADE 법칙이 적용된다.
  - ON DELETE 옵션 : 참조 테이블의 튜플이 삭제되었을 때 기본 테이블에 취해야 할 사항을 지정한다. 옵션에는 NO ACTION, CASCADE, SET NULL, SET DEFAULT가 있다.
  - ON UPDATE 옵션 : 참조 테이블의 참조 속성 값이 변경되었을 때 기본 테이블에 취해야 할 사항을 지정한다. 옵션에는 NO ACTION, CASCADE, SET NULL, SET DEFAULT가 있다.

- CONSTRAINT : 제약 조건의 이름을 지정한다. 이름을 지정할 필요가 없으면 CHECK절만 사용하여 속성 값에 대한 제약 조건을 명시한다.
- CHECK : 속성 값에 대한 제약 조건을 정의한다.

핵심 195 ALTER TABLE

ALTER TABLE은 테이블에 대한 정의를 변경하는 명령문이다.

표기 형식

ALTER TABLE 테이블명 **ADD** 속성명 데이터_타입 [DEFAULT '기본값'];

ALTER TABLE 테이블명 **ALTER** 속성명 [SET DEFAULT '기본값'];

ALTER TABLE 테이블명 **DROP COLUMN** 속성명 [CASCADE];

- ADD : 새로운 속성(열)을 추가할 때 사용한다.
- ALTER : 특정 속성의 Default 값을 변경할 때 사용한다.
- DROP COLUMN : 특정 속성을 삭제할 때 사용한다.

예제 1 〈학생〉 테이블에 최대 3문자로 구성되는 '학년' 속성 추가하시오.

```
ALTER TABLE 학생 ADD 학년 VARCHAR(3);
```

예제 2 〈학생〉 테이블의 '학번' 필드의 데이터 타입과 크기를 VARCHAR(10)으로 하고 NULL 값이 입력되지 않도록 변경하시오.

```
ALTER TABLE 학생 ALTER 학번 VARCHAR(10) NOT NULL;
```

핵심 196 DROP

DROP은 스키마, 도메인, 기본 테이블, 뷰 테이블, 인덱스, 제약 조건 등을 제거하는 명령문이다.

표기 형식

DROP SCHEMA 스키마명 [CASCADE | RESTRICT];

DROP DOMAIN 도메인명 [CASCADE | RESTRICT];

DROP TABLE 테이블명 [CASCADE | RESTRICT];

DROP VIEW 뷰명 [CASCADE | RESTRICT];

DROP INDEX 인덱스명 [CASCADE | RESTRICT];

DROP CONSTRAINT 제약조건명;

- CASCADE : 제거할 요소를 참조하는 다른 모든 개체를 함께 제거한다. 즉 주 테이블의 데이터 제거 시 각 외래키와 관계를 맺고 있는 모든 데이터를 제거하는 참조 무결성 제약 조건을 설정하기 위해 사용된다.
- RESTRICT : 다른 개체가 제거할 요소를 참조중일 때는 제거를 취소한다.

예제 〈학생〉 테이블을 제거하되, 〈학생〉 테이블을 참조하는 모든 데이터를 함께 제거하시오.

```
DROP TABLE 학생 CASCADE;
```

핵심 197 DCL(Data Control Language, 데이터 제어어)의 개념

DCL(데이터 제어어)는 데이터의 보안, 무결성, 회복, 병행 제어 등을 정의하는 데 사용하는 언어이다.

- DCL은 데이터베이스 관리자(DBA)가 데이터 관리를 목적으로 사용한다.
- DCL에는 GRANT, REVOKE, COMMIT, ROLLBACK, SAVEPOINT 등이 있다.

핵심 198 GRANT / REVOKE

데이터베이스 관리자가 데이터베이스 사용자에게 권한을 부여하거나 취소하기 위한 명령어이다.

- GRANT : 권한 부여를 위한 명령어
- REVOKE : 권한 취소를 위한 명령어
- 사용자등급 지정 및 해제

> – **GRANT** 사용자등급 **TO** 사용자_ID_리스트 [IDENTIFIED BY 암호];
> – **REVOKE** 사용자등급 **FROM** 사용자_ID_리스트;

예제 1 사용자 ID가 "NABI"인 사람에게 데이터베이스 및 테이블을 생성할 수 있는 권한을 부여하는 SQL문을 작성하시오.

```
GRANT RESOURCE TO NABI;
```

예제 2 사용자 ID가 "STAR"인 사람에게 단순히 데이터베이스에 있는 정보를 검색할 수 있는 권한을 부여하는 SQL문을 작성하시오.

```
GRANT CONNECT TO STAR;
```

- 테이블 및 속성에 대한 권한 부여 및 취소

> – **GRANT** 권한_리스트 **ON** 개체 **TO** 사용자 [WITH GRANT OPTION];
> – **REVOKE** [GRANT OPTION FOR] 권한_리스트 **ON** 개체 **FROM** 사용자 [CASCADE];

– **권한 종류** : ALL, SELECT, INSERT, DELETE, UPDATE, ALTER 등
– WITH GRANT OPTION : 부여받은 권한을 다른 사용자에게 다시 부여할 수 있는 권한을 부여함
– GRANT OPTION FOR : 다른 사용자에게 권한을 부여할 수 있는 권한을 취소함
– CASCADE : 권한 취소 시 권한을 부여받았던 사용자가 다른 사용자에게 부여한 권한도 연쇄적으로 취소함

예제 3 사용자 ID가 "NABI"인 사람에게 〈고객〉 테이블에 대한 모든 권한과 다른 사람에게 권한을 부여할 수 있는 권한까지 부여하는 SQL문을 작성하시오.

```
GRANT ALL ON 고객 TO NABI WITH GRANT OPTION;
```

예제 4 사용자 ID가 "STAR"인 사람에게 부여한 〈고객〉 테이블에 대한 권한 중 UPDATE 권한을 다른 사람에게 부여할 수 있는 권한만 취소하는 SQL문을 작성하시오.

```
REVOKE GRANT OPTION FOR UPDATE ON 고객 FROM STAR;
```

핵심 199 COMMIT

트랜잭션이 성공적으로 끝나면 데이터베이스가 새로운 일관성(Consistency) 상태를 가지기 위해 변경된 모든 내용을 데이터베이스에 반영하여야 하는데, 이때 사용하는 명령이 COMMIT이다.

- COMMIT 명령을 실행하지 않아도 DML문이 성공적으로 완료되면 자동으로 COMMIT되고, DML이 실패하면 자동으로 ROLLBACK이 되도록 Auto Commit 기능을 설정할 수 있다.

핵심 200 ROLLBACK

ROLLBACK은 아직 COMMIT되지 않은 변경된 모든 내용들을 취소하고 데이터베이스를 이전 상태로 되돌리는 명령어이다.

- 트랜잭션 전체가 성공적으로 끝나지 못하면 일부 변경된 내용만 데이터베이스에 반영되는 비일관성(Inconsistency)인 상태를 가질 수 있기 때문에 일부분만 완료된 트랜잭션은 롤백(Rollback)되어야 한다.

핵심 201 삽입문(INSERT INTO~)

삽입문은 기본 테이블에 새로운 튜플을 삽입할 때 사용한다.

일반 형식

```
INSERT INTO 테이블명([속성명1, 속성명2,…])
VALUES (데이터1, 데이터2,… );
```

- 대응하는 속성과 데이터는 개수와 데이터 유형이 일치해야 한다.
- 기본 테이블의 모든 속성을 사용할 때는 속성명을 생략할 수 있다.
- SELECT문을 사용하여 다른 테이블의 검색 결과를 삽입할 수 있다.

〈사원〉

이름	부서	생일	주소	기본급
홍길동	기획	04/05/61	망원동	120
임꺽정	인터넷	01/09/69	성산동	80
황진이	편집	07/21/75	연희동	100
김선달	편집	10/22/73	망원동	90
성춘향	기획	02/20/64	망원동	100
장길산	편집	03/11/67	상암동	120
일지매	기획	04/29/78	합정동	110
강호동	인터넷	12/11/80		90

예제 1 〈사원〉 테이블에 (이름 – 홍승현, 부서 – 인터넷)을 삽입하시오.

```
INSERT INTO 사원(이름, 부서) VALUES ('홍승현', '인터넷');
```

예제 2 〈사원〉 테이블에 (장보고, 기획, 05/03/73, 홍제동, 90)을 삽입하시오.

```
INSERT INTO 사원 VALUES ('장보고', '기획', #05/03/73#, '홍제동', 90);
```

예제 3 〈사원〉 테이블에 있는 편집부의 모든 튜플을 편집부원(이름, 생일, 주소, 기본급) 테이블에 삽입하시오.

```
INSERT INTO 편집부원(이름, 생일, 주소, 기본급)
SELECT 이름, 생일, 주소, 기본급
FROM 사원
WHERE 부서 = '편집';
```

핵심 202 삭제문(DELETE FROM~)

삭제문은 기본 테이블에 있는 튜플들 중에서 특정 튜플(행)을 삭제할 때 사용한다.

일반 형식

```
DELETE
FROM 테이블명
[WHERE 조건];
```

- 모든 레코드를 삭제할 때는 WHERE절을 생략한다.
- 모든 레코드를 삭제하더라도 테이블 구조는 남아 있기 때문에 디스크에서 테이블을 완전히 제거하는 DROP과는 다르다.

〈사원〉

이름	부서	생일	주소	기본급
홍길동	기획	04/05/61	망원동	120
임꺽정	인터넷	01/09/69	성산동	80
황진이	편집	07/21/75	연희동	100
김선달	편집	10/22/73	망원동	90
성춘향	기획	02/20/64	망원동	100
장길산	편집	03/11/67	상암동	120
일지매	기획	04/29/78	합정동	110
강호동	인터넷	12/11/80		90

예제 1 〈사원〉 테이블에서 "임꺽정"에 대한 튜플을 삭제하시오.

```
DELETE
FROM 사원
WHERE 이름 = '임꺽정';
```

핵심 203 갱신문(UPDATE~ SET~)

갱신문은 기본 테이블에 있는 튜플들 중에서 특정 튜플의 내용을 변경할 때 사용한다.

일반 형식

```
UPDATE 테이블명
SET 속성명 = 데이터[, 속성명=데이터, …]
[WHERE 조건];
```

〈사원〉

이름	부서	생일	주소	기본급
홍길동	기획	04/05/61	망원동	120
임꺽정	인터넷	01/09/69	성산동	80
황진이	편집	07/21/75	연희동	100
김선달	편집	10/22/73	망원동	90
성춘향	기획	02/20/64	망원동	100
장길산	편집	03/11/67	상암동	120
일지매	기획	04/29/78	합정동	110
강호동	인터넷	12/11/80		90

예제 〈사원〉 테이블에서 "황진이"의 '부서'를 "기획부"로 변경하고 '기본급'을 5만 원 인상시키시오.

```
UPDATE 사원
SET 부서 = '기획', 기본급 = 기본급 + 5
WHERE 이름 = '황진이';
```

핵심 204 데이터 조작문의 네 가지 유형

- SELECT(검색) : SELECT~ FROM~ WHERE~
- INSERT(삽입) : INSERT INTO~ VALUES~
- DELETE(삭제) : DELETE~ FROM~ WHERE~
- UPDATE(변경) : UPDATE~ SET~ WHERE~

핵심 205 일반 형식

```
SELECT [PREDICATE] [테이블명.]속성명 [AS 별칭][, [테이블명.]속성명, …]
[, 그룹함수(속성명) [AS 별칭]]
[, Window함수 OVER (PARTITION BY 속성명1, 속성명2, …
                    ORDER BY 속성명3, 속성명4, …)]
FROM 테이블명[, 테이블명, …]
[WHERE 조건]
[GROUP BY 속성명, 속성명, …]
[HAVING 조건]
[ORDER BY 속성명 [ASC | DESC]];
```

- SELECT절
  - PREDICATE : 불러올 튜플 수를 제한할 명령어를 기술한다.
    - ▶ ALL : 모든 튜플을 검색할 때 지정하는 것으로, 주로 생략한다.
    - ▶ DISTINCT : 중복된 튜플이 있으면 그 중 첫 번째 한 개만 검색한다.
    - ▶ DISTINCTROW : 중복된 튜플을 제거하고 한 개만 검색하지만 선택된 속성의 값이 아닌, 튜플 전체를 대상으로 한다.
  - 속성명 : 검색하여 불러올 속성(열) 또는 속성을 이용한 수식을 지정한다.
    - ▶ 기본 테이블을 구성하는 모든 속성을 지정할 때는 '*'를 기술한다.
    - ▶ 두 개 이상의 테이블을 대상으로 검색할 때는 '테이블명.속성명'으로 표현한다.
  - AS : 속성 및 연산의 이름을 다른 제목으로 표시하기 위해 사용된다.
- FROM절 : 질의에 의해 검색될 데이터들을 포함하는 테이블명을 기술한다.
- WHERE절 : 검색할 조건을 기술한다.
- ORDER BY절 : 특정 속성을 기준으로 정렬하여 검색할 때 사용한다.

– 속성명 : 정렬의 기준이 되는 속성명을 기술한다.

– [ASC | DESC] : 정렬 방식으로서 'ASC'는 오름차순, 'DESC'는 내림차순이다. 생략하면 오름차순으로 지정된다.

〈사원〉

이름	부서	생일	주소	기본급
홍길동	기획	04/05/61	망원동	120
임꺽정	인터넷	01/09/69	서교동	80
황진이	편집	07/21/75	합정동	100
김선달	편집	10/22/73	망원동	90
성춘향	기획	02/20/64	대흥동	100
장길산	편집	03/11/67	상암동	120
일지매	기획	04/29/78	연남동	110
강건달	인터넷	12/11/80		90

〈여가활동〉

이름	취미	경력
김선달	당구	10
성춘향	나이트댄스	5
일지매	태껸	15
임꺽정	씨름	8

예제 1 〈사원〉 테이블에서 '주소'만 검색하되 같은 '주소'는 한 번만 출력하시오.

```
SELECT DISTINCT 주소
FROM 사원;
```

〈결과〉

주소
대흥동
망원동
상암동
서교동
연남동
합정동

예제 2 〈사원〉 테이블에서 "기획" 부서에 근무하면서 "대흥동"에 사는 사람의 튜플을 검색하시오.

```
SELECT *
FROM 사원
WHERE 부서 = '기획' AND 주소 = '대흥동';
```

〈결과〉

이름	부서	생일	주소	기본급
성춘향	기획	02/20/64	대흥동	100

핵심 206 조건 연산자 / 연산자 우선순위

조건 연산자

- 비교 연산자

연산자	=	〈〉	〉	〈	〉=	〈=
의미	같다	같지 않다	크다	작다	크거나 같다	작거나 같다

- 논리 연산자 : NOT, AND, OR
- LIKE 연산자 : 대표 문자를 이용해 지정된 속성의 값이 문자 패턴과 일치하는 튜플을 검색하기 위해 사용된다.

대표 문자	%	_	#
의미	모든 문자를 대표함	문자 하나를 대표함	숫자 하나를 대표함

연산자 우선순위

종류	연산자	우선순위
산술 연산자	×, /, +, –	왼쪽에서 오른쪽으로 갈수록 낮아진다.
관계 연산자	=, 〈〉, 〉, 〉=, 〈, 〈=	모두 같다.
논리 연산자	NOT, AND, OR	왼쪽에서 오른쪽으로 갈수록 낮아진다.

※ 산술, 관계, 논리 연산자가 함께 사용되었을 때는 산술 〉 관계 〉 논리 연산자 순서로 연산자 우선순위가 정해진다.

〈사원〉

이름	부서	생일	주소	기본급
홍길동	기획	04/05/61	망원동	120
임꺽정	인터넷	01/09/69	서교동	80
황진이	편집	07/21/75	합정동	100
김선달	편집	10/22/73	망원동	90
성춘향	기획	02/20/64	대흥동	100
장길산	편집	03/11/67	상암동	120
일지매	기획	04/29/78	연남동	110
강건달	인터넷	12/11/80		90

〈여가활동〉

이름	취미	경력
김선달	당구	10
성춘향	나이트댄스	5
일지매	태껸	15
임꺽정	씨름	8

예제 〈사원〉 테이블에서 성이 '김'인 사람의 튜플을 검색하시오.

```
SELECT *
FROM 사원
WHERE 이름 LIKE "김%";
```

〈결과〉

이름	부서	생일	주소	기본급
김선달	편집	10/22/73	망원동	90

핵심 207 하위 질의

하위 질의는 조건절에 주어진 질의를 먼저 수행하여 그 검색 결과를 조건절의 피연산자로 사용한다.

〈사원〉

이름	부서	생일	주소	기본급
홍길동	기획	04/05/61	망원동	120
임꺽정	인터넷	01/09/69	서교동	80
황진이	편집	07/21/75	합정동	100
김선달	편집	10/22/73	망원동	90
성춘향	기획	02/20/64	대흥동	100
장길산	편집	03/11/67	상암동	120
일지매	기획	04/29/78	연남동	110
강건달	인터넷	12/11/80		90

〈여가활동〉

이름	취미	경력
김선달	당구	10
성춘향	나이트댄스	5
일지매	태껸	15
임꺽정	씨름	8

예제 1 '취미'가 "나이트댄스"인 사원의 '이름'과 '주소'를 검색하시오.

```
SELECT 이름, 주소
FROM 사원
WHERE 이름 = (SELECT 이름 FROM 여가활동
WHERE 취미 = '나이트댄스') ;
```

〈결과〉

이름	주소
성춘향	대흥동

예제 2 취미활동을 하는 사원들의 부서를 검색하시오.

```
SELECT 부서
FROM 사원
WHERE EXISTS (SELECT 이름 FROM 여가활동
WHERE 여가활동.이름 = 사원.이름);
```

〈결과〉

부서
인터넷
편집
기획
기획

핵심 208 일반 형식

```
SELECT [PREDICATE] [테이블명.]속성명 [AS 별칭]
[, [테이블명.]속성명, …]
[, 그룹함수(속성명) [AS 별칭]]
[, WINDOW함수 OVER (PARTITION BY 속성명1,
속성명2, …
                    ORDER BY 속성명3, 속성명4, …)
[AS 별칭]]
FROM 테이블명[, 테이블명, …]
[WHERE 조건]
[GROUP BY 속성명, 속성명, …]
[HAVING 조건]
[ORDER BY 속성명 [ASC | DESC]];
```

- **그룹함수** : GROUP BY절에 지정된 그룹별로 속성의 값을 집계할 함수를 기술한다.
- WINDOW 함수 : GROUP BY절을 이용하지 않고 속성의 값을 집계할 함수를 기술한다.
  - PARTITION BY : WINDOW 함수가 적용될 범위로 사용할 속성을 지정한다.
  - ORDER BY : PARTITION 안에서 정렬 기준으로 사용할 속성을 지정한다.
- GROUP BY절 : 특정 속성을 기준으로 그룹화하여 검색할 때 사용한다. 일반적으로 GROUP BY절은 그룹 함수와 함께 사용된다.
- HAVING절 : GROUP BY와 함께 사용되며, 그룹에 대한 조건을 지정한다.

〈상여금〉

부서	이름	상여내역	상여금
기획	홍길동	연장근무	100
기획	일지매	연장근무	100
기획	최준호	야간근무	120
기획	장길산	특별근무	90
인터넷	강건달	특별근무	90
인터넷	서국현	특별근무	90
인터넷	박인식	연장근무	30
편집	김선달	특별근무	80
편집	황종근	연장근무	40
편집	성춘향	야간근무	80
편집	임꺽정	야간근무	80
편집	황진이	야간근무	50

예제 〈상여금〉 테이블에서 '상여금'이 100 이상인 사원이 2명 이상인 '부서'의 튜플 수를 구하시오.

```
SELECT 부서, COUNT(*) AS 사원수
FROM 상여금
WHERE 상여금 >= 100
GROUP BY 부서
HAVING COUNT(*) >= 2;
```

〈결과〉

부서	사원수
기획	3

핵심 209 그룹 함수

GROUP BY절에 지정된 그룹별로 속성의 값을 집계할 때 사용된다.

- COUNT(속성명) : 그룹별 튜플 수를 구하는 함수
- SUM(속성명) : 그룹별 합계를 구하는 함수
- AVG(속성명) : 그룹별 평균을 구하는 함수
- MAX(속성명) : 그룹별 최대값을 구하는 함수
- MIN(속성명) : 그룹별 최소값을 구하는 함수
- STDDEV(속성명) : 그룹별 표준편차를 구하는 함수
- VARIANCE(속성명) : 그룹별 분산을 구하는 함수

핵심 210 집합 연산자를 이용한 통합 질의

집합 연산자를 사용하여 2개 이상의 테이블의 데이터를 하나로 통합한다.

표기 형식

```
SELECT 속성명1, 속성명2, …
FROM 테이블명
UNION | UNION ALL | INTERSECT | EXCEPT
SELECT 속성명1, 속성명2, …
FROM 테이블명
[ORDER BY 속성명 [ASC | DESC]];
```

- 두 개의 SELECT문에 기술한 속성들은 개수와 데이터 유형이 서로 동일해야 한다.

〈사원〉

사원	직급
김형석	대리
홍영선	과장
류기선	부장
김현천	이사

〈직원〉

사원	직급
신원섭	이사
이성호	대리
홍영선	과장
류기선	부장

예제 〈사원〉 테이블과 〈직원〉 테이블을 통합하는 질의문을 작성하시오. (단, 같은 레코드가 중복되어 나오지 않게 하시오.)

```
SELECT *
FROM 사원
UNION
SELECT *
FROM 직원;
```

〈결과〉

사원	직급
김현천	이사
김형석	대리
류기선	부장
신원섭	이사
이성호	대리
홍영선	과장

핵심 211 JOIN의 개념

JOIN(조인)은 2개의 테이블에 대해 연관된 튜플들을 결합하여, 하나의 새로운 릴레이션을 반환한다.

- JOIN은 크게 INNER JOIN과 OUTER JOIN으로 구분된다.
- JOIN은 일반적으로 FROM절에 기술하지만, 릴레이션이 사용되는 어느 곳에서나 사용할 수 있다.

핵심 212 INNER JOIN

INNER JOIN은 일반적으로 EQUI JOIN과 NON-EQUI JOIN으로 구분된다.

- 조건이 없는 INNER JOIN을 수행하면 CROSS JOIN과 동일한 결과를 얻을 수 있다.

- EQUI JOIN
  - EQUI JOIN은 JOIN 대상 테이블에서 공통 속성을 기준으로 '='(equal) 비교에 의해 같은 값을 가지는 행을 연결하여 결과를 생성하는 JOIN 방법이다.
  - WHERE절을 이용한 EQUI JOIN의 표기 형식

```
SELECT [테이블명1.]속성명, [테이블명2.]속성명, …
FROM 테이블명1, 테이블명2, …
WHERE 테이블명1.속성명 = 테이블명2.속성명;
```

  - NATURAL JOIN절을 이용한 EQUI JOIN의 표기 형식

```
SELECT [테이블명1.]속성명, [테이블명2.]속성명, …
FROM 테이블명1 NATURAL JOIN 테이블명2;
```

  - JOIN ~ USING절을 이용한 EQUI JOIN의 표기 형식

```
SELECT [테이블명1.]속성명, [테이블명2.]속성명, …
FROM 테이블명1 JOIN 테이블명2 USING(속성명);
```

〈학생〉

학번	이름	학과코드	선배	성적
15	고길동	com		83
16	이순신	han		96
17	김선달	com	15	95
19	아무개	han	16	75
37	박치민		17	55

〈학과〉

학과코드	학과명
com	컴퓨터
han	국어
eng	영어

〈성적등급〉

등급	최저	최고
A	90	100
B	80	89
C	60	79
D	0	59

예제 1 〈학생〉 테이블과 〈학과〉 테이블에서 '학과코드' 값이 같은 튜플을 JOIN하여 '학번', '이름', '학과코드', '학과명'을 출력하는 SQL문을 작성하시오.

```
SELECT 학번, 이름, 학생.학과코드, 학과명
FROM 학생, 학과
WHERE 학생.학과코드 = 학과.학과코드;
```

```
SELECT 학번, 이름, 학생.학과코드, 학과명
FROM 학생 NATURAL JOIN 학과;
```

```
SELECT 학번, 이름, 학생.학과코드, 학과명
FROM 학생 JOIN 학과 USING(학과코드);
```

〈결과〉

학번	이름	학과코드	학과명
15	고길동	com	컴퓨터
16	이순신	han	국어
17	김선달	com	컴퓨터
19	아무개	han	국어

핵심 213 SELF JOIN

- SELF JOIN은 같은 테이블에서 2개의 속성을 연결하여 EQUI JOIN을 하는 JOIN 방법이다.
- 표기 형식

```
SELECT [별칭1.]속성명, [별칭1.]속성명, …
FROM 테이블명1 [AS] 별칭1 JOIN 테이블명1 [AS] 별칭2
ON 별칭1.속성명 = 별칭2.속성명;
```

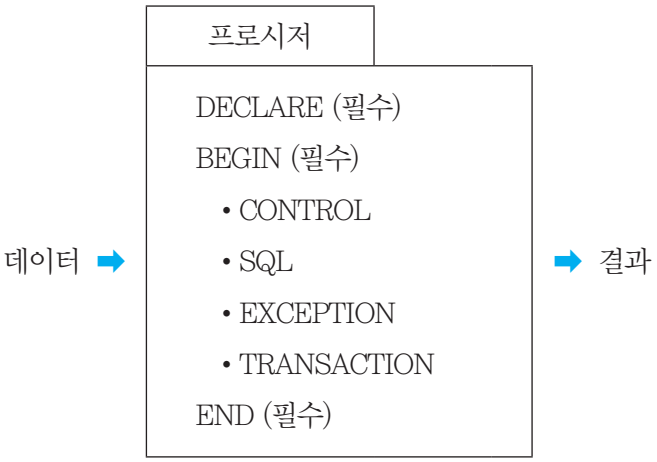
```
SELECT [별칭1.]속성명, [별칭1.]속성명, …
FROM 테이블명1 [AS] 별칭1, 테이블명1 [AS] 별칭2
WHERE 별칭1.속성명 = 별칭2.속성명;
```

핵심 214 프로시저(Procedure)의 개요

프로시저란 절차형 SQL을 활용하여 특정 기능을 수행하는 일종의 트랜잭션 언어로, 호출을 통해 실행되어 미리 저장해 놓은 SQL 작업을 수행한다.

- 프로시저를 만들어 데이터베이스에 저장하면 여러 프로그램에서 호출하여 사용할 수 있다.
- 프로시저는 데이터베이스에 저장되어 수행되기 때문에 스토어드(Stored) 프로시저라고도 불린다.
- 프로시저는 시스템의 일일 마감 작업, 일괄(Batch) 작업 등에 주로 사용된다.

프로시저의 구성도



- DECLARE : 프로시저의 명칭, 변수, 인수, 데이터 타입을 정의하는 선언부이다.
- BEGIN / END : 프로시저의 시작과 종료를 의미한다.
- CONTROL : 조건문 또는 반복문이 삽입되어 순차적으로 처리된다.
- SQL : DML, DCL이 삽입되어 데이터 관리를 위한 조회, 추가, 수정, 삭제 작업을 수행한다.
- EXCEPTION : BEGIN ~ END 안의 구문 실행 시 예외가 발생하면 이를 처리하는 방법을 정의한다.
- TRANSACTION : 수행된 데이터 작업들을 DB에 적용할지 취소할지를 결정하는 처리부이다.

핵심 215 트리거(Trigger)의 개요

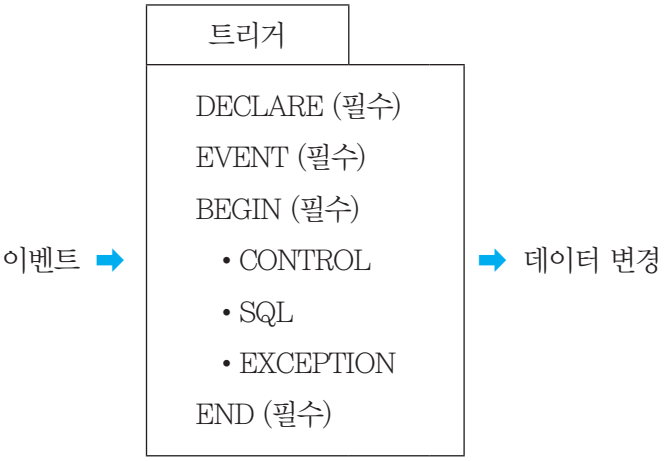
트리거는 데이터베이스 시스템에서 데이터의 삽입(Insert), 갱신(Update), 삭제(Delete) 등의 이벤트(Event)가 발생할 때마다 관련 작업이 자동으로 수행되는 절차형 SQL이다.

- 트리거는 데이터베이스에 저장되며, 데이터 변경 및 무결성 유지, 로그 메시지 출력 등의 목적으로 사용된다.
- 트리거의 구문에는 DCL(데이터 제어어)을 사용할 수 없으며, DCL이 포함된 프로시저나 함수를 호출하는 경우에도 오류가 발생한다.
- 트리거에 오류가 있는 경우 트리거가 처리하는 데이터에도 영향을 미치므로 트리거를 생성할 때 세심한 주의가 필요하다.

핵심 216 트리거의 구성

트리거는 선언, 이벤트, 시작, 종료로 구성되며, 시작과 종료 구문 사이에는 제어(CONTROL), SQL, 예외(EXCEPTION)가 포함된다.

트리거 구성도



- DECLARE : 트리거의 명칭, 변수 및 상수, 데이터 타입을 정의하는 선언부이다.
- EVENT : 트리거가 실행되는 조건을 명시한다.
- BEGIN / END : 트리거의 시작과 종료를 의미한다.
- CONTROL : 조건문 또는 반복문이 삽입되어 순차적으로 처리된다.

- SQL : DML문이 삽입되어 데이터 관리를 위한 조회, 추가, 수정, 삭제 작업을 수행한다.
- EXCEPTION : BEGIN ~ END 안의 구문 실행 시 예외가 발생하면 이를 처리하는 방법을 정의한다.

핵심 217 사용자 정의 함수

사용자 정의 함수의 개요

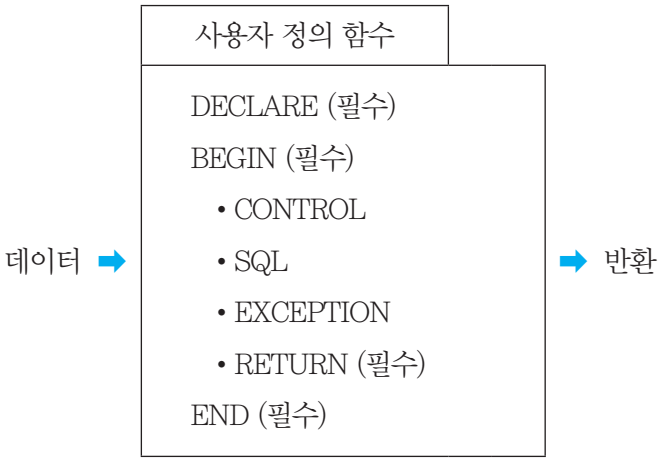
사용자 정의 함수는 프로시저와 유사하게 SQL을 사용하여 일련의 작업을 연속적으로 처리하며, 종료 시 처리 결과를 단일값으로 반환하는 절차형 SQL이다.

- 사용자 정의 함수는 데이터베이스에 저장되어 SELECT, INSERT, DELETE, UPDATE 등 DML문의 호출에 의해 실행된다.
- 사용자 정의 함수는 예약어 RETURN을 통해 값을 반환하기 때문에 출력 파라미터가 없다.
- 사용자 정의 함수는 INSERT, DELETE, UPDATE를 통한 테이블 조작은 할 수 없고 SELECT를 통한 조회만 할 수 있다.
- 사용자 정의 함수는 프로시저를 호출하여 사용할 수 없다.
- 사용자 정의 함수는 SUM( ), AVG( ) 등의 내장 함수처럼 DML문에서 반환값을 활용하기 위한 용도로 사용된다.

사용자 정의 함수의 구성

사용자 정의 함수의 구성은 프로시저와 유사하다. 프로시저의 구성에서 RETURN만 추가하면 된다.

- 사용자 정의 함수의 구성도

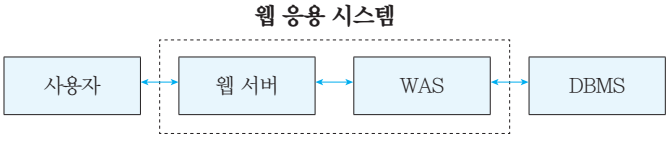


- DECLARE : 사용자 정의 함수의 명칭, 변수, 인수, 데이터 타입을 정의하는 선언부이다.
- BEGIN / END : 사용자 정의 함수의 시작과 종료를 의미한다.
- CONTROL : 조건문 또는 반복문이 삽입되어 순차적으로 처리된다.
- SQL : SELECT문이 삽입되어 데이터 조회 작업을 수행한다.
- EXCEPTION : BEGIN ~ END 안의 구문 실행 시 예외가 발생하면 이를 처리하는 방법을 정의한다.
- RETURN : 호출 프로그램에 반환할 값이나 변수를 정의한다.

핵심 218 DBMS 접속의 개요

DBMS 접속은 사용자가 데이터를 사용하기 위해 응용 시스템을 이용하여 DBMS에 접근하는 것을 의미한다.

- 응용 시스템은 사용자로부터 매개 변수를 전달받아 SQL을 실행하고 DBMS로부터 전달받은 결과를 사용자에게 전달하는 매개체 역할을 수행한다.
- 인터넷을 통해 구동되는 웹 응용 프로그램은 웹 응용 시스템을 통해 DBMS에 접근한다.
- 웹 응용 시스템은 웹 서버와 웹 애플리케이션 서버(WAS)로 구성되며, 서비스 규모가 작은 경우 웹 서버와 웹 애플리케이션 서버를 통합하여 하나의 서버만으로 운용할 수 있다.



- 사용자는 웹 서버에 접속하여 데이터를 주고받는다.
- 웹 서버는 많은 수의 서비스 요청을 처리하기 때문에 사용자가 대용량의 데이터를 요청하면 직접 처리하지 않고 WAS에게 해당 요청을 전달한다.
- WAS는 수신한 요청을 트랜잭션 언어로 변환한 후 DBMS에 전달하여 데이터를 받는다. 이렇게 받은 데이터는 처음 요청한 웹 서버로 다시 전달되어 사용자에게까지 도달하게 된다.

핵심 219 DBMS 접속 기술

DBMS 접속 기술은 DBMS에 접근하기 위해 사용하는 API 또는 API의 사용을 편리하게 도와주는 프레임워크 등을 의미한다.

JDBC(Java DataBase Connectivity)

JDBC는 Java 언어로 다양한 종류의 데이터베이스에 접속하고 SQL문을 수행할 때 사용되는 표준 API이다.

- 1997년 2월 썬 마이크로시스템에서 출시했다.
- 접속하려는 DBMS에 대한 드라이버가 필요하다.

ODBC(Open DataBase Connectivity)

ODBC는 데이터베이스에 접근하기 위한 표준 개방형 API로, 개발 언어에 관계없이 사용할 수 있다.

- 1992년 9월 마이크로소프트에서 출시했다.
- ODBC도 접속하려는 DBMS에 맞는 드라이버가 필요하지만, 접속하려는 DBMS의 인터페이스를 알지 못하더라도 ODBC 문장을 사용하여 SQL을 작성하면 ODBC에 포함된 드라이버 관리자가 해당 DBMS의 인터페이스에 맞게 연결해 주므로 DBMS의 종류를 몰라도 된다.

MyBatis

MyBatis는 JDBC 코드를 단순화하여 사용할 수 있는 SQL Mapping 기반 오픈 소스 접속 프레임워크이다.

- JDBC로 데이터베이스에 접속하려면 다양한 메소드를 호출하고 해제해야 하는데, MyBatis는 이를 간소화 했고 접속 기능을 더욱 강화하였다.
- MyBatis는 SQL 문장을 분리하여 XML 파일을 만들고, Mapping을 통해 SQL을 실행한다.
- MyBatis는 SQL을 거의 그대로 사용할 수 있어 SQL 친화적인 국내 환경에 적합하여 많이 사용된다.

핵심 220 동적 SQL(Dynamic SQL)

동적 SQL은 개발 언어에 삽입되는 SQL 코드를 문자열 변수에 넣어 처리하는 것으로, 조건에 따라 SQL 구문을 동적으로 변경하여 처리할 수 있다.

- 동적 SQL은 사용자로부터 SQL문의 일부 또는 전부를 입력받아 실행할 수 있다.
- 동적 SQL은 값이 입력되지 않을 경우 사용하는 NVL 함수를 사용할 필요가 없다.
- 동적 SQL은 응용 프로그램 수행 시 SQL이 변형될 수 있으므로 프리컴파일 할 때 구문 분석, 접근 권한 확인 등을 할 수 없다.
- 동적 SQL은 정적 SQL에 비해 속도가 느리지만, 상황에 따라 다양한 조건을 첨가하는 등 유연한 개발이 가능하다.

핵심 221 SQL 테스트의 개요

SQL 테스트는 SQL이 작성 의도에 맞게 원하는 기능을 수행하는지 검증하는 과정이다.

- 단문 SQL은 SQL 코드를 직접 실행한 후 결과를 확인하는 것으로 간단히 테스트가 가능하다.
- 절차형 SQL은 테스트 전에 생성을 통해 구문 오류(Syntax Error)나 참조 오류의 존재 여부를 확인한다.
- 정상적으로 생성된 절차형 SQL은 디버깅을 통해 로직을 검증하고, 결과를 통해 최종적으로 확인한다.

핵심 222 단문 SQL 테스트

단문 SQL 테스트는 DDL, DML, DCL이 포함되어 있는 SQL과 TCL을 테스트하는 것으로, 직접 실행하여 결과물을 확인한다.

- 실행 시 오류나 경고가 발생할 경우 메시지를 참조하여 문제를 해결한다.
- DESCRIBE 명령어를 이용하면 DDL로 작성된 테이블이나 뷰의 속성, 자료형, 옵션들을 바로 확인할 수 있다.
  - DESC [개체명];
- DML로 변경한 데이터는 SELECT문으로 데이터의 정상적인 변경 여부를 확인할 수 있다.
- DCL로 설정된 사용자 권한은 사용자 권한 정보가 저장된 테이블을 SELECT로 조회하거나, SHOW 명령어로 확인할 수 있다.

핵심 223 절차형 SQL 테스트

프로시저, 사용자 정의 함수, 트리거 등의 절차형 SQL은 디버깅을 통해 기능의 적합성 여부를 검증하고, 실행을 통해 결과를 확인하는 테스트를 수행한다.

- 많은 코드로 구성된 절차형 SQL의 특성상 오류 및 경고 메시지가 상세히 출력되지 않으므로 SHOW 명령어를 통해 오류 내용을 확인하고 문제를 수정한다.

  – 형식 : SHOW ERRORS;

- 데이터베이스에 변화를 줄 수 있는 SQL문은 주석으로 처리하고, 출력문을 이용하여 화면에 출력하여 확인한다.
- 디버깅이 완료되면 출력문을 삭제하고, 주석 기호를 삭제한 후 절차형 SQL을 실행하여 결과를 검토한다.

핵심 224 ORM(Object-Relational Mapping)의 개요

ORM은 객체지향 프로그래밍의 객체(Object)와 관계형 데이터베이스(Relational Database)의 데이터를 연결(Mapping)하는 기술을 의미한다.

- ORM은 객체지향 프로그래밍에서 사용할 수 있는 가상의 객체지향 데이터베이스를 만들어 프로그래밍 코드와 데이터를 연결한다.
- ORM으로 생성된 가상의 객체지향 데이터베이스는 프로그래밍 코드 또는 데이터베이스와 독립적이므로 재사용 및 유지보수가 용이하다.
- ORM은 SQL 코드를 직접 입력하지 않고 선언문이나 할당 같은 부수적인 코드가 생략되기 때문에 직관적이고 간단하게 데이터를 조작할 수 있다.

핵심 225 ORM 프레임워크

ORM 프레임워크는 ORM을 구현하기 위한 구조와 구현을 위해 필요한 여러 기능들을 제공하는 소프트웨어를 의미한다.

- ORM 프레임워크의 종류

JAVA	JPA, Hibernate, EclipseLink, DataNucleus, Ebean 등
C++	ODB, QxOrm 등
Python	Django, SQLAlchemy, Storm 등
iOS	DatabaseObjects, Core Data 등
.NET	NHibernate, DatabaseObjects, Dapper 등
PHP	Doctrine, Propel, RedBean 등

핵심 226 쿼리 성능 최적화의 개요

쿼리 성능 최적화는 데이터 입 · 출력 애플리케이션의 성능 향상을 위해 SQL 코드를 최적화하는 것이다.

- 쿼리 성능을 최적화하기 전에 성능 측정 도구인 APM을 사용하여 최적화 할 쿼리를 선정해야 한다.
- 최적화 할 쿼리에 대해 옵티마이저가 수립한 실행 계획을 검토하고 SQL 코드와 인덱스를 재구성한다.

※ RBO vs CBO

RBO(Rule Based Optimizer)는 규칙 기반 옵티마이저이고, CBO(Cost Based Optimizer)는 비용 기반 옵티마이저로서 다음과 같은 차이점이 있다.

	RBO	CBO
최적화 기준	규칙에 정의된 우선순위	액세스 비용
성능 기준	개발자의 SQL 숙련도	옵티마이저의 예측 성능
특징	실행 계획 예측이 쉬움	성능 통계치 정보 활용, 예측이 복잡함
고려사항	개발자의 규칙 이해도, 규칙의 효율성	비용 산출 공식의 정확성

핵심 227 쿼리 성능 최적화

쿼리 성능 최적화는 실행 계획에 표시된 연산 순서, 조인 방식, 테이블 조회 방법 등을 참고하여 SQL문이 더 빠르고 효율적으로 작동하도록 SQL 코드와 인덱스를 재구성하는 것을 의미한다.

SQL 코드 재구성

- WHERE 절을 추가하여 일부 레코드만 조회하게 함으로써 조회에 들어가는 비용을 줄인다.
- WHERE 절에 연산자가 포함되면 INDEX를 활용하지 못하므로 가능한 한 연산자 사용을 자제한다.
- 서브 쿼리에 특정 데이터가 존재하는지 확인할 때는 IN보다 EXISTS를 활용한다.
- 옵티마이저의 실행 계획이 잘못되었다고 판단되는 경우 힌트를 활용하여 실행 계획의 액세스 경로 및 조인 순서를 변경한다.

인덱스 재구성

- SQL 코드에서 조회되는 속성과 조건들을 고려하여 인덱스를 구성한다.
- 실행 계획을 참고하여 인덱스를 추가하거나 기존 인덱스의 열 순서를 변경한다.
- 인덱스의 추가 및 변경은 해당 테이블을 참조하는 다른 SQL문에도 영향을 줄 수 있으므로 신중히 결정한다.
- 단일 인덱스로 쓰기나 수정 없이 읽기로만 사용되는 테이블의 경우 IOT(Index-Organized Table)로 구성하는 것을 고려한다.
- 불필요한 인덱스를 제거한다.

핵심 228 데이터 전환의 정의

데이터 전환이란 운영 중인 기존 정보 시스템에 축적되어 있는 데이터를 추출(Extraction)하여 새로 개발할 정보 시스템에서 운영 가능하도록 변환(Transformation)한 후, 적재(Loading)하는 일련의 과정을 말한다.

- 데이터 전환을 ETL(Extraction, Transformation, Load), 즉 추출, 변환, 적재 과정이라고 한다.
- 데이터 전환을 데이터 이행(Data Migration) 또는 데이터 이관이라고도 한다.

핵심 229 데이터 검증

데이터 검증이란 원천 시스템의 데이터를 목적 시스템의 데이터로 전환하는 과정이 정상적으로 수행되었는지 여부를 확인하는 과정을 말한다.

- 데이터 전환 검증은 검증 방법과 검증 단계에 따라 분류할 수 있다.

핵심 230 검증 방법에 따른 분류

데이터 검증은 검증 방법에 따라 로그 검증, 기본 항목 검증, 응용 프로그램 검증, 응용 데이터 검증, 값 검증으로 분류할 수 있다.

로그 검증	데이터 전환 과정에서 작성하는 추출, 전환, 적재 로그를 검증한다.
기본 항목 검증	로그 검증 외에 별도로 요청된 검증 항목에 대해 검증한다.
응용 프로그램 검증	응용 프로그램을 통한 데이터 전환의 정합성을 검증한다.
응용 데이터 검증	사전에 정의된 업무 규칙을 기준으로 데이터 전환의 정합성을 검증한다.
값 검증	숫자 항목의 합계 검증, 코드 데이터의 범위 검증, 속성 변경에 따른 값 검증을 수행한다.

핵심 231 오류 데이터 측정 및 정제의 개요

오류 데이터 측정 및 정제는 고품질의 데이터를 운영 및 관리하기 위해 수행한다.

- 오류 데이터 측정 및 정제는 '데이터 품질 분석 → 오류 데이터 측정 → 오류 데이터 정제' 순으로 진행한다.

핵심 232 오류 데이터 정제

오류 데이터 정제는 오류 관리 목록의 각 항목을 분석하여 원천 데이터를 정제하거나 전환 프로그램을 수정하는 것이다.

오류 데이터 분석

- 오류 관리 목록의 오류 데이터를 분석하여 오류 상태, 심각도, 해결 방안을 확인 및 기재한다.
- 상태

Open	오류가 보고만 되고 분석되지 않은 상태
Assigned	오류의 영향 분석 및 수정을 위해 개발자에게 오류를 전달한 상태
Fixed	개발자가 오류를 수정한 상태
Closed	수정된 오류에 대해 테스트를 다시 했을 때 오류가 발견되지 않은 상태
Deferred	오류 수정을 연기한 상태
Classified	보고된 오류를 관련자들이 확인했을 때 오류가 아니라고 확인된 상태

핵심 233 데이터 정제요청서의 개요

데이터 정제요청서는 원천 데이터의 정제와 전환 프로그램의 수정을 위해 요청사항 및 조치사항 등 데이터 정제와 관련된 전반적인 내용을 문서로 작성한 것이다.

- 오류 관리 목록을 기반으로 데이터 정제 요건 목록을 작성하고 이 목록의 항목별로 데이터 정제요청서를 작성한다.

핵심 234 데이터 정제보고서의 개요 및 작성

데이터 정제보고서는 데이터 정제요청서를 통해 정제된 원천 데이터가 정상적으로 정제되었는지 확인한 결과를 문서로 작성한 것이다.

- 정제 요청 데이터와 정제된 데이터 항목을 육안으로 직접 비교하여 확인한다.
- 정제된 데이터를 데이터 전환 프로그램을 이용하여 전환한 후 오류의 발생 여부를 확인하고 목적 데이터베이스에서 전환된 데이터를 확인한다.
- 정제된 데이터의 확인 결과를 반영한 데이터 정제보고서를 정제ID별로 작성한다.
- 데이터 정제보고서에는 데이터 전환 결과 외에도 오류 데이터의 원인, 실제 데이터 정제 건수, 향후 대응 방안 등을 포함한다.

4 과목 프로그래밍 언어 활용

핵심 235 하드웨어 환경

하드웨어 환경은 사용자와의 인터페이스 역할을 하는 클라이언트(Client) 그리고 클라이언트와 통신하여 서비스를 제공하는 서버(Server)로 구성된다.

- 클라이언트에는 PC, 스마트폰 등이 있다.
- 서버는 사용 목적에 따라 웹 서버, 웹 애플리케이션 서버, 데이터베이스 서버, 파일 서버 등으로 나뉜다.
  - 웹 서버(Web Server) : 클라이언트로부터 직접 요청을 받아 처리하는 서버로, 저용량의 정적 파일들을 제공한다.
  - 웹 애플리케이션 서버(WAS; Web Application Server) : 사용자에게 동적 서비스를 제공하기 위해 웹 서버로부터 요청을 받아 데이터 가공 작업을 수행하거나, 웹 서버와 데이터베이스 서버 또는 웹 서버와 파일 서버 사이에서 인터페이스 역할을 수행하는 서버이다.
  - 데이터베이스 서버(DB Server) : 데이터베이스와 이를 관리하는 DBMS를 운영하는 서버이다.
  - 파일 서버(File Server) : 데이터베이스에 저장하기에는 비효율적이거나, 서비스 제공을 목적으로 유지하는 파일들을 저장하는 서버이다.

핵심 236 소프트웨어 환경

소프트웨어 환경은 클라이언트와 서버 운영을 위한 시스템 소프트웨어와 개발에 사용되는 개발 소프트웨어로 구성된다.

- 시스템 소프트웨어에는 운영체제(OS), 웹 서버 및 WAS 운용을 위한 서버 프로그램, DBMS 등이 있다.
- 개발 소프트웨어에는 요구사항 관리 도구, 설계/모델링 도구, 구현 도구, 빌드 도구, 테스트 도구, 형상 관리 도구 등이 있다.
  - 요구사항 관리 도구 : 요구사항의 수집과 분석, 추적 등을 편리하게 도와주는 소프트웨어
  - 설계/모델링 도구 : UML(통합 모델링 언어)을 지원하며, 개발의 전 과정에서 설계 및 모델링을 도와주는 소프트웨어
  - 구현 도구 : 개발 언어를 통해 애플리케이션의 실제 구현을 지원하는 소프트웨어
  - 빌드 도구 : 구현 도구를 통해 작성된 소스의 빌드 및 배포, 라이브러리 관리를 지원하는 소프트웨어
  - 테스트 도구 : 모듈들이 요구사항에 적합하게 구현되었는지 테스트하는 소프트웨어
  - 형상 관리 도구 : 산출물들을 버전별로 관리하여 품질 향상을 지원하는 소프트웨어

핵심 237 서버 개발 프레임워크

서버 개발 프레임워크는 서버 프로그램 개발 시 다양한 네트워크 설정, 요청 및 응답 처리, 아키텍처 모델 구현 등을 손쉽게 처리할 수 있도록 클래스나 인터페이스를 제공하는 소프트웨어를 의미한다.

- 서버 개발 프레임워크에 따라 지원하는 프로그래밍 언어가 제한적이므로 선정할 수 있는 프레임워크도 제한적이다.
- 서버 개발 프레임워크의 대부분은 모델-뷰-컨트롤러(MVC) 패턴을 기반으로 개발되었다.
- 대표적인 서버 개발 프레임워크의 종류는 다음과 같다.

프레임워크	특징
Spring	JAVA를 기반으로 만들어진 프레임워크로, 전자정부 표준 프레임워크의 기반 기술로 사용되고 있다.
Node.js	JavaScript를 기반으로 만들어진 프레임워크로, 비동기 입·출력 처리와 이벤트 위주의 높은 처리 성능을 갖고 있어 실시간으로 입·출력이 빈번한 애플리케이션에 적합하다.
Django	Python을 기반으로 만들어진 프레임워크로, 컴포넌트의 재사용과 플러그인화를 강조하여 신속한 개발이 가능하도록 지원한다.
Codeigniter	PHP를 기반으로 만들어진 프레임워크로, 인터페이스가 간편하며 서버 자원을 적게 사용한다.
Ruby on Rails	Ruby를 기반으로 만들어진 프레임워크로, 테스트를 위한 웹 서버를 지원하며 데이터베이스 작업을 단순화, 자동화시켜 개발 코드의 길이가 짧아 신속한 개발이 가능하다.

핵심 238 소프트웨어 개발 보안 점검 항목

소프트웨어 개발 보안 점검 항목은 소프트웨어 개발의 각 단계에서 점검되어야 할 보안 항목들을 말한다.

세션 통제	세션은 서버와 클라이언트의 연결을 말하며, 세션 통제는 세션의 연결과 연결로 인해 발생하는 정보를 관리하는 것을 의미한다.
입력 데이터 검증 및 표현	입력 데이터에 대한 유효성 검증체계를 갖추고, 검증 실패 시 이를 처리할 수 있도록 코딩하는 것을 의미한다.
보안 기능	인증, 접근제어, 기밀성, 암호화 등의 기능을 의미한다.
시간 및 상태	동시 수행을 지원하는 병렬 처리 시스템이나 다수의 프로세스가 동작하는 환경에서 시간과 실행 상태를 관리하여 시스템이 원활히 동작되도록 코딩하는 것을 의미한다.
에러처리	• 소프트웨어 실행 중 발생할 수 있는 오류들을 사전에 정의하여 에러로 인해 발생할 수 있는 문제들을 예방하는 것을 의미한다. • 보안 약점에는 오류 메시지를 통한 정보 노출, 오류 상황 대응 부재 등이 있다.
코드 오류	개발자들이 코딩 중 실수하기 쉬운 형(Type) 변환, 자원의 반환 등을 고려하며 코딩하는 것을 의미한다.
캡슐화	데이터(속성)와 데이터를 처리하는 함수를 하나의 객체로 묶어 코딩하는 것을 의미한다.
API 오용	API를 잘못 사용하거나 보안에 취약한 API를 사용하지 않도록 고려하여 코딩하는 것을 의미한다.

핵심 239 API(Application Programming Interface)

API는 응용 프로그램 개발 시 운영체제나 프로그래밍 언어 등에 있는 라이브러리를 이용할 수 있도록 규칙 등을 정의해 놓은 인터페이스를 의미한다.

- API는 프로그래밍 언어에서 특정한 작업을 수행하기 위해 사용되거나, 운영체제의 파일 제어, 화상 처리, 문자 제어 등의 기능을 활용하기 위해 사용된다.
- API는 개발에 필요한 여러 도구를 제공하기 때문에 이를 이용하면 원하는 기능을 쉽고 효율적으로 구현할 수 있다.
- API의 종류에는 Windows API, 단일 유닉스 규격(SUS), Java API, 웹 API 등이 있으며, 누구나 무료로 사용할 수 있게 공개된 API를 Open API라고 한다.

핵심 240 배치 프로그램

배치 프로그램(Batch Program)의 개요

배치 프로그램은 사용자와의 상호 작용 없이 여러 작업들을 미리 정해진 일련의 순서에 따라 일괄적으로 처리하는 것을 의미한다.

- 배치 프로그램이 갖추어야 하는 필수 요소는 다음과 같다.

대용량 데이터	대량의 데이터를 가져오거나, 전달하거나, 계산하는 등의 처리가 가능해야 한다.
자동화	심각한 오류가 발생하는 상황을 제외하고는 사용자의 개입 없이 수행되어야 한다.
견고성	잘못된 데이터나 데이터 중복 등의 상황으로 중단되는 일 없이 수행되어야 한다.
안정성/신뢰성	오류가 발생하면 오류의 발생 위치, 시간 등을 추적할 수 있어야 한다.
성능	다른 응용 프로그램의 수행을 방해하지 않아야 하고, 지정된 시간 내에 처리가 완료되어야 한다.

핵심 241 패키지 소프트웨어

패키지 소프트웨어는 기업에서 일반적으로 사용하는 여러 기능들을 통합하여 제공하는 소프트웨어를 의미한다.

- 기업에서는 패키지 소프트웨어를 구입하여 기업 환경에 적합하게 커스터마이징(Customizing)하여 사용한다.
- 패키지 소프트웨어를 이용하여 시스템을 구축하는 방식을 패키지 개발 방식이라고 한다.
- 기능 요구사항을 70% 이상 충족시키는 패키지 소프트웨어가 있을 때만 사용하는 것이 적합하다.

※ 패키지 소프트웨어와 전용 개발 소프트웨어의 비교

	패키지 소프트웨어	전용 개발 소프트웨어
기능 요구사항	70% 이상 충족시키는 패키지 소프트웨어가 있는 경우 이용	모든 기능 요구사항 반영 가능
안정성	품질이 검증되었고, 업계 표준 준용	개발자의 역량에 따라 달라짐
라이선스	판매자	회사
생산성	개발을 위한 인력과 시간이 절약됨	개발을 위한 인력과 시간이 필요함
호환성	보장이 안 됨	설계 단계부터 고려하여 개발
유지보수	결함 발생 시 즉시 대응이 어려움	결함 발생 시 즉시 대응이 가능

불합격 방지용 안전장치 기억상자

틀린 문제만 모아 오답 노트를 만들고 싶다고요?
까먹기 전에 다시 한 번 복습하고 싶다고요?
지금 당장 QR 코드를 스캔해 보세요.

핵심 242 C/C++의 데이터 타입 크기 및 기억 범위

종류	데이터 타입	크기
문자	char	1Byte
부호없는 문자형	unsigned char	1Byte
정수	short	2Byte
	int	4Byte
	long	4Byte
	long long	8Byte
실수	float	4Byte
	double	8Byte
	long double	8Byte

핵심 243 C언어의 구조체

배열이 자료의 형과 크기가 동일한 변수의 모임이라면 구조체는 자료의 종류가 다른 변수의 모임이라고 할 수 있다. 예를 들어 이름, 직위, 급여 등의 필드가 필요한 사원 자료를 하나의 단위로 관리하려면 이름과 직위는 문자, 급여는 숫자와 같이 문자와 숫자가 혼용되므로 배열로는 처리할 수 없다. 이런 경우 구조체를 사용하면 간단하게 처리할 수 있다.

- 구조체를 정의한다는 것은 int나 char 같은 자료형을 하나 만드는 것을 의미한다.
- 구조체는 'structure(구조)'의 약어인 'struct'를 사용하여 정의한다.
- 구조체 정의 예

```
struct sawon {
   char name[10];
   char position[10];
   int pay;
}
```

핵심 244 JAVA의 데이터 타입 크기 및 기억 범위

종류	데이터 타입	크기
문자	char	2Byte
정수	byte	1Byte
	short	2Byte
	int	4Byte
	long	8Byte
실수	float	4Byte
	double	8Byte
논리	boolean	1Byte

핵심 245 변수의 개요 / 변수명 작성 규칙

변수의 개요

변수(Variable)는 컴퓨터가 명령을 처리하는 도중 발생하는 값을 저장하기 위한 공간으로, 변할 수 있는 값을 의미한다.

- 변수는 저장하는 값에 따라 정수형, 실수형, 문자형, 포인터형 등으로 구분한다.

변수명 작성 규칙

- 영문자, 숫자, _(under bar)를 사용할 수 있다.
- 첫 글자는 영문자나 _(under bar)로 시작해야 하며, 숫자는 올 수 없다.
- 글자 수에 제한이 없다.
- 공백이나 *, +, -, / 등의 특수문자를 사용할 수 없다.
- 대 · 소문자를 구분한다.
- 예약어를 변수명으로 사용할 수 없다.
- 변수 선언 시 문장 끝에 반드시 세미콜론(;)을 붙여야 한다.
- 변수 선언 시 변수명에 데이터 타입을 명시하는 것을 헝가리안 표기법(Hungarian Notation)이라고 한다.

자동 변수(Automatic Variable)

자동 변수는 함수나 코드의 범위를 한정하는 블록 내에서 선언되는 변수이다.

- 함수나 블록이 실행되는 동안에만 존재하며 이를 벗어나면 자동으 소멸된다.

외부 변수(External Variable)

외부 변수는 현재 파일이나 다른 파일에서 선언된 변수나 함수를 참조(reference)하기 위한 변수이다.

- 외부 변수는 함수 밖에서 선언한다.

정적 변수(Static Variable)

정적 변수는 함수나 블록 내에서 선언하는 내부 정적 변수와 함수 외부에서 선언하는 외부 정적 변수가 있다.

- 내부 정적 변수는 선언한 함수나 블록 내에서만 사용할 수 있고, 외부 정적 변수는 모든 함수에서 사용할 수 있다.

레지스터 변수(Register Variable)

레지스터 변수는 메모리가 아닌 CPU 내부의 레지스터에 기억영역을 할당받는 변수이다.

- 자주 사용되는 변수를 레지스터에 저장하여 처리 속도를 높이기 위해 사용한다.

핵심 246 기억 클래스

변수 선언 시 메모리 내에 변수의 값을 저장하기 위한 기억영역이 할당되는데, 할당되는 기억영역에 따라 사용범위에 제한이 있다. 이러한 기억영역을 결정하는 작업을 기억 클래스(Storage Class)라 한다.

- C언어에서는 다음과 같이 5가지 종류의 기억 클래스를 제공한다.

종류	기억영역	예약어	생존기간	사용 범위
자동 변수	메모리(스택)	auto	일시적	지역적
레지스터 변수	레지스터	register		
정적 변수(내부)	메모리(데이터)	static	영구적	
정적 변수(외부)				전역적
외부 변수		extern		

핵심 247 연산자의 종류

산술 연산자

산술 연산자는 가, 감, 승, 제 등의 산술 계산에 사용되는 연산자를 말한다.

- 산술 연산자에는 일반 산술식과 달리 한 변수의 값을 증가하거나 감소시키는 증감 연산자가 있다.

연산자	의미	비고
+	덧셈	
-	뺄셈	
*	곱셈	
/	나눗셈	

%	나머지	
++	증가 연산자	전치 : 변수 앞에 증감 연산자가 오는 형태로 먼저 변수의 값을 증감시킨 후 변수를 연산에 사용한다(++a, --a). 후치 : 변수 뒤에 증감 연산자가 오는 형태로 먼저 변수를 연산에 사용한 후 변수의 값을 증감시킨다(a++, a--).
--	감소 연산자	

관계 연산자

관계 연산자는 두 수의 관계를 비교하여 참(true) 또는 거짓(false)을 결과로 얻는 연산자이다.

- 거짓은 0, 참은 1로 사용되지만 0외의 모든 숫자도 참으로 간주된다.

연산자	의미
==	같다
!=	같지 않다
>	크다
>=	크거나 같다
<	작다
<=	작거나 같다

비트 연산자

비트 연산자는 비트별(0, 1)로 연산하여 결과를 얻는 연산자이다.

연산자	의미	비고
&	and	모든 비트가 1일 때만 1
^	xor	모든 비트가 같으면 0, 하나라도 다르면 1
|	or	모든 비트 중 한 비트라도 1이면 1
~	not	각 비트의 부정, 0이면 1, 1이면 0
<<	왼쪽 시프트	비트를 왼쪽으로 이동
>>	오른쪽 시프트	비트를 오른쪽으로 이동

논리 연산자

논리 연산자는 두 개의 논리 값을 연산하여 참(true) 또는 거짓(false)을 결과로 얻는 연산자이다. 관계 연산자와 마찬가지로 거짓은 0, 참은 1이다.

연산자	의미	비고
!	not	부정
&&	and	모두 참이면 참
||	or	하나라도 참이면 참

대입 연산자

연산 후 결과를 대입하는 연산식을 간략하게 입력할 수 있도록 대입 연산자를 제공한다. 대입 연산자는 산술, 관계, 비트, 논리 연산자에 모두 적용할 수 있다.

연산자	예	의미
+=	a += 1	a = a + 1
-=	a -= 1	a = a - 1
*=	a *= 1	a = a * 1
/=	a /= 1	a = a / 1
%=	a %= 1	a = a % 1
<<=	a <<= 1	a = a << 1
>>=	a >>= 1	a = a >> 1

조건 연산자

조건 연산자는 조건에 따라 서로 다른 수식을 수행한다.

형식

조건 ? 수식1 : 수식2;	'조건'의 수식이 참이면 '수식1'을, 거짓이면 '수식2'를 실행한다.

기타 연산자

연산자	의미
sizeof	자료형의 크기를 표시한다.
,(콤마)	콤마로 구분하여 한 줄에 두 개 이상의 수식을 작성하거나 변수를 정의한다.
(자료형)	• 사용자가 자료형을 다른 자료형으로 변환할 때 사용하는 것으로, cast(캐스트) 연산자라고 부른다. • 변환할 자료형을 괄호 안에 넣어서 변환할 값이나 변수명 앞에 놓는다.

핵심 248 연산자 우선순위

- 한 개의 수식에 여러 개의 연산자가 사용되면 기본적으로 아래 표의 순서대로 처리된다.
- 아래 표의 한 줄에 가로로 나열된 연산자는 우선순위가 같기 때문에 결합규칙에 따라 ←는 오른쪽에 있는 연산자부터, →는 왼쪽에 있는 연산자부터 차례로 계산된다.

대분류	중분류	연산자	결합 규칙	우선 순위
단항 연산자	단항 연산자	!(논리 not) ~(비트 not) ++(증가) --(감소) sizeof(기타)	←	높음
이항 연산자	산술 연산자	* / %(나머지)	→	↑
		+ -		
	시프트 연산자	<< >>		
	관계 연산자	< <= >= >		
		==(같다) !=(같지 않다)		
	비트 연산자	&(비트 and) ^(비트 xor) |(비트 or)		
	논리 연산자	&&(논리 and) ||(논리 or)		
삼항 연산자	조건 연산자	? :	→	↓
대입 연산자	대입 연산자	= += -= *= /= %= <<= >>= 등	←	
순서 연산자	순서 연산자	,	→	낮음

핵심 249 scanf( ) 함수

scanf( ) 함수는 C언어의 표준 입력 함수로, 키보드로 입력받아 변수에 저장하는 함수이다.

형식

scanf (서식 문자열, 변수의 주소)	• 서식 문자열 : 입력받을 데이터의 자료형을 지정한다. • 변수의 주소 : 데이터를 입력받을 변수를 적는다. 변수의 주소로 입력받아야 하기 때문에 변수에 주소연산자 &를 붙인다.

예 scanf("%3d", &a);

▶ % : 서식 문자임을 지정

▶ 3 : 입력 자릿수를 3자리로 지정

▶ d : 10진수로 입력

▶ &a : 입력받은 데이터를 변수 a의 주소에 저장

특징

- 입력받을 데이터의 자료형, 자릿수 등을 지정할 수 있다.
- 한 번에 여러 개의 데이터를 입력 받을 수 있다.
- 서식 문자열과 변수의 자료형은 일치해야 한다.

  예 scanf("%d %f", &i, &j); → '%d'와 i, "%f"와 j는 자료형이 일치해야 한다.

서식 문자열

서식 문자열은 printf( ) 함수로 출력할 때도 동일하게 적용된다.

서식 문자열	의미
%d	정수형 10진수를 입 · 출력하기 위해 지정한다.
%u	부호없는 정수형 10진수를 입 · 출력하기 위해 지정한다.
%o	정수형 8진수를 입 · 출력하기 위해 지정한다.
%x	정수형 16진수를 입 · 출력하기 위해 지정한다.
%c	문자를 입 · 출력하기 위해 지정한다.
%s	문자열을 입 · 출력하기 위해 지정한다.
%f	소수점을 포함하는 실수를 입 · 출력하기 위해 지정한다.
%e	지수형 실수를 입 · 출력하기 위해 지정한다.
%ld	long형 10진수를 입 · 출력하기 위해 지정한다.
%lo	long형 8진수를 입 · 출력하기 위해 지정한다.
%lx	long형 16진수를 입 · 출력하기 위해 지정한다.
%p	주소를 16진수로 입 · 출력하기 위해 지정한다.

핵심 250 printf( ) 함수

printf( ) 함수는 C언어의 표준 출력 함수로, 인수로 주어진 값을 화면에 출력하는 함수이다.

형식

printf (서식 문자열, 변수)	• 서식 문자열 : 변수의 자료형에 맞는 서식 문자열을 입력한다. • 변수 : 서식 문자열의 순서에 맞게 출력할 변수를 적는다. scanf( )와 달리 주소 연산자 &를 붙이지 않는다.

예 printf("%-8.2f", 200.2);
(∨는 빈 칸을 의미함)

```
200.20∨∨
```

▶ % : 서식 문자임을 지정

▶ - : 왼쪽부터 출력

▶ 8 : 출력 자릿수를 8자리로 지정

▶ 2 : 소수점 이하를 2자리로 지정

▶ f : 실수로 출력

주요 제어문자

제어문자란 입력 혹은 출력 내용을 제어하는 문자이다.

문자	의미	기능
\n	new line	커서를 다음 줄 앞으로 이동한다.
\b	backspace	커서를 왼쪽으로 한 칸 이동한다.
\t	tab	커서를 일정 간격 띄운다.
\r	carriage return	커서를 현재 줄의 처음으로 이동한다.
\0	null	널 문자를 출력한다.
\'	single quote	작은따옴표를 출력한다.
\"	double quote	큰따옴표를 출력한다.
\a	alert	스피커로 벨 소리를 출력한다.
\\	backslash	역 슬래시를 출력한다.
\f	form feed	한 페이지를 넘긴다.

핵심 251 JAVA에서의 표준 출력

• JAVA에서 값을 화면에 출력할 때는 System 클래스의 서브 클래스인 out 클래스의 메소드 print( ), println( ), printf( ) 등을 사용하여 출력한다.

• 형식 1 : 서식 문자열에 맞게 변수의 내용을 출력한다.

```
System.out.printf(서식 문자열, 변수)
```

– printf( ) 메소드는 C언어의 printf( ) 함수와 사용법이 동일하다.

예 System.out.printf("%-8.2f", 200.2);
(V는 빈 칸을 의미함)

```
200.20VV
```

▶ % : 서식 문자임을 지정

▶ - : 왼쪽부터 출력

▶ 8 : 출력 자릿수를 8자리로 지정

▶ 2 : 소수점 이하를 2자리로 지정

▶ f : 실수로 출력

• 형식 2 : 값이나 변수의 내용을 형식없이 출력한다.

```
System.out.print( )
```

– 문자열을 출력할 때는 큰따옴표로 묶어줘야 한다.

– 문자열 또는 문자열 변수를 연속으로 출력할 때는 +를 이용한다.

예 System.out.print("abc123" + "def");

```
abc123def
```

• 형식 3 : 값이나 변수의 내용을 형식없이 출력한 후 커서를 다음 줄의 처음으로 이동한다.

```
System.out.println( )
```

– println( ) 메소드는 출력 후 다음 줄로 이동한다는 것을 제외하면 print( ) 메소드와 사용법이 동일하다.

예 System.out.print("abc123" + "def");

```
abc123def
▮
```

커서의 위치

핵심 252 기타 표준 입·출력 함수

입력	getchar( )	키보드로 한 문자를 입력받아 변수에 저장하는 함수
	gets( )	키보드로 문자열을 입력받아 변수에 저장하는 함수로, Enter를 누르기 전까지를 하나의 문자열로 인식하여 저장함
출력	putchar( )	인수로 주어진 한 문자를 화면에 출력하는 함수
	puts( )	인수로 주어진 문자열을 화면에 출력한 후 커서를 자동으로 다음 줄 앞으로 이동하는 함수

핵심 253 단순 if문

if문은 조건에 따라서 실행할 문장을 달리하는 제어문이며, 단순 if문은 조건이 한 개 일 때 사용하는 제어문이다.

- 조건이 참일 때만 실행할 문장을 지정할 수도 있고, 참과 거짓에 대해 각각 다른 실행문을 지정할 수도 있다.
- 형식1 : 조건이 참일 때만 실행한다.
  - 조건이 참일 때 실행할 문장이 하나인 경우

```
if(조건)
    실행할 문장;
```

if는 조건 판단문에 사용되는 예약어이므로 그대로 적는다.
조건은 참(1) 또는 거짓(0)이 결과로 나올 수 있는 수식을 ( ) 안에 입력한다.
조건이 참일 경우 실행할 문장을 적는다.

  - 조건이 참일 때 실행할 문장이 두 문장 이상인 경우

```
if(조건)
{
    실행할 문장1;
    실행할 문장2;
     :
}
```

{ } 사이에 조건이 참일 경우 실행할 문장을 적는다.

예제 1 a가 10보다 크면 a에서 10을 빼기

```
#include <stdio.h>
main( )
{
  int a = 15, b;
  if (a > 10) ❶
     b = a - 10; ❷
  printf("%d\n", b); ❸
```

a가 10보다 크면 ❷번 문장을 실행하고, 아니면 ❸번 문장으로 이동해서 실행을 계속한다.
❶번의 조건식이 참일 경우 실행할 문장이다. b는 5가 된다.
여기서는 ❶번의 조건식이 거짓일 경우 실행할 문장이 없다. 조건 판단문을 벗어나면 무조건 ❸번으로 온다.

결과 5

형식2 : 조건이 참일 때와 거짓 때 실행할 문장이 다르다.

```
if(조건)
   실행할 문장1;
else
   실행할 문장2;
```

조건이 참일 경우 실행할 문장을 적는다. 참일 경우 실행할 문장이 두 문장 이상이면 { }를 입력하고 그 사이에 문장을 적는다.

조건이 거짓일 경우 실행할 문장을 적는다. 두 문장 이상인 경우 {}를 입력하고 그 사이에 문장을 적는다.

예제 2 a가 b보다 크면 'a-b', 아니면 'b-a'를 수행하기

```
#include <stdio.h>
main( )
{
  int a = 10, b = 20, cha;
  if (a > b) ❶
     cha = a - b; ❷
  else ❸
     cha = b - a; ❹
  printf("%d\n", cha);
}
```

a가 b보다 크면 ❷번 문장을 실행하고, 아니면 ❸번의 다음 문장인 ❹번 문장을 실행한다.

❶번의 조건식이 참일 경우 실행할 문장이다. 참이 아니기 때문에 초기화 시키지 않은 cha에는 알 수 없는 값이 그대로 있게 된다.

❶번의 조건식이 거짓일 경우 실행할 문장의 시작점이다.

❶번의 조건식이 거짓일 경우 실행할 실제 처리문이다. cha는 10이 된다.

결과 10

핵심 254 switch문

switch문은 조건에 따라 분기할 곳이 여러 곳인 경우 간단하게 처리할 수 있는 제어문이다.

• 형식

```
switch(수식) ❶
{ ❷
  case 레이블1: ❸
     실행할 문장1;
     break;
  case 레이블2: ❹
     실행할 문장2;
     break;
       :
```

• switch는 switch문에 사용되는 예약어로 그대로 입력한다.
• 수식 : '레이블1' ~ '레이블n'의 값 중 하나를 도출하는 변수나 수식을 입력한다.

❷~❺번이 switch문의 범위이다. '{'로 시작해서 '}'로 끝난다. 반드시 입력해야 한다.

• case는 switch문에서 레이블을 지정하기 위한 예약어로 그대로 입력해야 한다.
• 레이블1 : ❶번 식의 결과가 될 만한 값 중 하나를 입력한다. 결과가 '레이블1'과 일치하면 이곳으로 찾아온다. 식의 결과가 5종류로 나타나면 case문이 5번 나와야한다.

❶번 식의 결과가 ❸번의 '레이블1'과 일치할 때 실행할 문장이다.

switch문을 탈출하여 ❺번으로 간다.

❶번의 식의 결과가 '레이블2'와 일치하면 찾아오는 곳이다.

❶번의 식의 결과가 ❹번의 '레이블2'와 일치할 때 실행할 문장이다.

switch문을 탈출하여 ❺번으로 간다.

```
  default:                    ❶번의 식의 결과가 '레이블1' ~ '레이블n'에 해당하지 않는 경우 찾아오는 곳이다.
    실행할 문장3;
} ❺
```

- case문의 레이블에는 한 개의 상수만 지정할 수 있으며, int, char, enum형의 상수만 가능하다.
- case문의 레이블에는 변수를 지정할 수 없다.
- break문은 생략이 가능하지만 break문이 생략되면 수식과 레이블이 일치할 때 실행할 문장부터 break문 또는 switch문이 종료될 때까지 모든 문장이 실행된다.

예제 점수(jum)에 따라 등급 표시하기

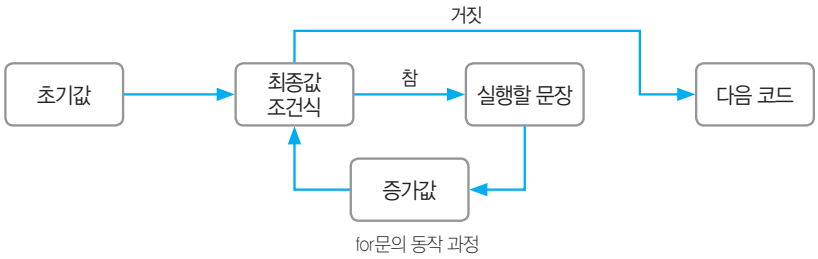
```
#include <stdio.h>
main( )
{
  int jum = 85;
  switch (jum / 10)           jum을 10으로 나눠 결과에 해당하는 숫자를 찾아간다. 85/10은 8.5지만 C 언어에서 정수 나눗셈은 결과도 정수이므로 결과는 8이다. 8에 해당하는 ❺번으로 이동하여 ❻, ❼번을 실행한다.
  { ❶                         ❶~❽번까지가 switch 조건문의 범위이다.
    case 10:                  100점일 경우 'jum/10'의 결과인 10이 찾아오는 곳이지만 할 일은 'case 9:'와 같으므로 아무것도 적지 않는다. 아무것도 적지 않으면 다음 문장인 ❷번으로 이동한다.
    case 9: ❷                 'jum/10'이 9일 경우 찾아오는 곳이다. ❸, ❹번을 실행한다.
      printf("학점은 A입니다.\n"); ❸   "학점은 A입니다."를 출력한다.
      break; ❹                break를 만나면 switch문을 탈출하여 ❾번으로 이동한다.
    case 8: ❺                 'jum/10'이 8일 경우 찾아오는 곳이다. ❻, ❼번을 실행한다.
      printf("학점은 B입니다.\n"); ❻   "학점은 B입니다."를 출력한다.
      break; ❼                switch문을 탈출하여 ❾번으로 이동한다.
    case 7:
      printf("학점은 C입니다.\n");
      break;
    case 6:
      printf("학점은 D입니다.\n");
      break;
    default:                  case 10~6에 해당되지 않는 경우, 즉 jum이 59 이하인 경우 찾아오는 곳이다.
      printf("학점은 F입니다.\n");   "학점은 F입니다."를 출력한다.
    } ❽
} ❾
```

결과 **학점은 B입니다.**

핵심 255 for문

for문은 초기값, 최종값, 증가값을 지정하는 수식을 이용해 정해진 횟수를 반복하는 제어문이다.

- for문은 초기값을 정한 다음 최종값에 대한 조건이 참이면 실행할 문장을 실행한 후 초기값을 증가값 만큼 증가시키면서 최종값에 대한 조건이 참인 동안 실행할 문장을 반복 수행한다.



for문의 동작 과정

- 형식

for(식1; 식2; 식3)
실행할 문장;

- for는 반복문을 의미하는 예약어로 그대로 입력한다.
- 식1 : 초기값을 지정할 수식을 입력한다.
- 식2 : 최종값을 지정할 수식을 입력한다.
- 식3 : 증가값으로 사용할 수식을 입력한다.

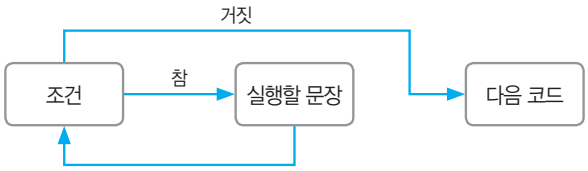
식2가 참일 동안 실행할 문장을 입력한다. 실행할 문장이 두 문장 이상일 경우 { }를 입력하고 그 사이에 처리할 문장들을 입력한다.

- for문은 처음부터 최종값에 대한 조건식을 만족하지 못하면 한 번도 수행하지 않는다.

핵심 256 while문

while문은 조건이 참인 동안 실행할 문장을 반복 수행하는 제어문이다.

- while문은 조건이 참인동안 실행할 문장을 반복 수행하다가 조건이 거짓이면 while문을 끝낸 후 다음 코드를 실행한다.
- while문은 조건이 처음부터 거짓이면 한 번도 수행하지 않는다.



while문의 동작 과정

- 형식

while(조건)
실행할 문장;

- while은 반복문에 사용되는 예약어로 그대로 입력한다.
- (조건) : 참이나 거짓을 결과로 갖는 수식을 '조건'에 입력한다. 참(1)을 직접 입력할 수도 있다.

조건이 참인 동안 실행할 문장을 입력한다. 문장이 두 문장 이상인 경우 { }를 입력하고 그 사이에 처리할 문장들을 입력한다.

예제 다음은 1~5까지의 합을 더하는 프로그램이다. 결과를 확인하시오.

```
#include <stdio.h>
main( )
{
   int a = 0, hap = 0;
   while (a < 5) ❶
   {  ❷
      a++; ❸
      hap += a; ❹
   } ❺
   printf("%d, %d\n", a, hap); ❻
}
```

a가 5보다 작은 동안 ❷~❺번 문장을 반복하여 수행한다.

❷~❺번까지가 반복문의 범위이다. 반복문에서 실행할 문장이 하나인 경우는 { }를 생략해도 된다.

'a = a + 1;'과 동일하다. a의 값을 1씩 증가시킨다.

'hap = hap + a'와 동일하다. a의 값을 hap에 누적시킨다.

반복문의 끝이다.

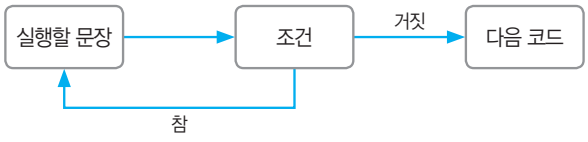
결과 5, 15

a가 5가 되었을 때 5를 hap에 누적한 다음 while 문을 벗어나기 때문에 a는 5로 끝난다.

핵심 257 do~while문

do~while문은 조건이 참인 동안 정해진 문장을 반복 수행하다가 조건이 거짓이면 반복문을 벗어나는 while문과 같은 동작을 하는데, 다른 점은 do~while문은 실행할 문장을 무조건 한 번 실행한 다음 조건을 판단하여 탈출 여부를 결정한다는 것이다.

- do~while문은 실행할 문장을 우선 실행한 후 조건을 판별하여 조건이 참이면 실행할 문장을 계속 반복 수행하고, 거짓이면 do~while문을 끝낸 후 다음 코드를 실행한다.



do~while문의 동작 과정

- 형식

```
do
   실행할 문장;
while(조건);
```

do는 do~while문에 사용되는 예약어로, do~while의 시작 부분에 그대로 입력한다.

조건이 참인 동안 실행할 문장을 입력한다. 문장이 두 문장 이상인 경우 { }를 입력하고 그 사이에 실행할 문장들을 입력한다.

- while은 do~while문에 사용되는 예약어로, do~while의 끝 부분에 그대로 입력한다.
- (조건) : 참이나 거짓을 결과로 갖는 수식을 '조건'에 입력한다. 참(1)을 직접 입력할 수도 있다.

예제 다음은 1부터 10까지 홀수의 합을 더하는 프로그램이다. 결과를 확인하시오.

```
#include <stdio.h>
main ( )
{
   int a = 1, hap = 0;
   do ❶
   { ❷
      hap += a; ❸
      a += 2; ❹
   } while(a < 10); ❺
   printf("%d, %d\n", a, hap); ❻

}
```

do~while 반복문의 시작점이다. ❷~❺번 사이의 문장을 반복하여 수행한다.

❷~❺번까지가 반복문의 범위이다.

'hap = hap + a'와 동일하다. a의 값을 hap에 누적시킨다.

'a = a + 2;'와 동일하다. a의 값을 2씩 증가시킨다.

a가 10보다 작은 동안 ❷~❺번 사이의 문장을 반복하여 수행한다.

결과 **11, 25**

a가 9가 되었을 때 9를 hap에 누적한 다음 a에 2를 더해 a가 11이 되었을 때 do~while문을 벗어나기 때문에 a는 11로 끝난다.

핵심 258 배열

배열의 개념

배열은 동일한 데이터 유형을 여러 개 사용해야 할 경우 이를 손쉽게 처리하기 위해 여러 개의 변수들을 조합해서 하나의 이름으로 정의해 사용하는 것을 말한다.

- 배열은 하나의 이름으로 여러 기억장소를 가리키기 때문에 배열에서 개별적인 요소들의 위치는 첨자를 이용하여 지정한다.
- 배열은 변수명 뒤에 대괄호 [ ]를 붙이고 그 안에 사용할 개수를 지정한다.
- C언어에서 배열의 위치는 0부터 시작된다.
- 배열은 행 우선으로 데이터가 기억장소에 할당된다.
- C 언어에서 배열 위치를 나타내는 첨자 없이 배열 이름을 사용하면 배열의 첫 번째 요소의 주소를 지정하는 것과 같다.

1차원 배열

- 1차원 배열은 변수들을 일직선상의 개념으로 조합한 배열이다.
- 형식

자료형 변수명[개수];	• 자료형 : 배열에 저장할 자료의 형을 지정한다. • 변수명 : 사용할 배열의 이름으로 사용자가 임의로 지정한다. • 개수 : 배열의 크기를 지정하는 것으로 생략할 수 있다.

예 int a[5] : 5개의 요소를 갖는 정수형 배열 a

	첫 번째	두 번째	세 번째	네 번째	다섯 번째
배열 a	a[0]	a[1]	a[2]	a[3]	a[4]

※ a[3] : a는 배열의 이름이고, 3은 첨자로서 배열 a에서의 위치를 나타낸다. a[3]에 4를 저장시키려면 'a[3] = 4'와 같이 작성한다.

예제 1 1차원 배열 a의 각 요소에 10, 11, 12, 13, 14를 저장한 후 출력하기

```
#include <stdio.h>
main( )
{
  int a[5];

  int i;
  for (i = 0; i < 5; i++)
    a[i] = i + 10; ❶

  for (i = 0; i < 5; i++)
    printf("%d ", a[i]); ❷

}
```

5개의 요소를 갖는 정수형 배열 a를 선언한다. 선언할 때는 사용할 개수를 선언하고, 사용할 때는 첨자를 0부터 사용하므로 주의해야 한다.

	첫 번째	두 번째	세 번째	네 번째	다섯 번째
배열 a	a[0]	a[1]	a[2]	a[3]	a[4]

정수형 변수 i를 선언한다

반복 변수 i가 0에서 시작하여 1씩 증가하면서 5보다 작은 동안 ❶번 문장을 반복하여 수행한다. 그러니까 ❶번 문장을 5회 반복하는 것이다.

배열 a의 i번째에 i+10을 저장시킨다. i는 0~4까지 변하므로 배열 a에 저장된 값은 다음과 같다.

배열 a	10	11	12	13	14
	a[0]	a[1]	a[2]	a[3]	a[4]

반복 변수 i가 0에서 시작하여 1씩 증가하면서 5보다 작은 동안 ❷번 문장을 반복하여 수행한다.

배열 a의 i번째를 출력한다. i는 0~4까지 변하므로 출력 결과는 다음과 같다. 서식 문자열에 '\n'이 없기 때문에 한 줄에 붙여서 출력한다.

결과 `10 11 12 13 14`

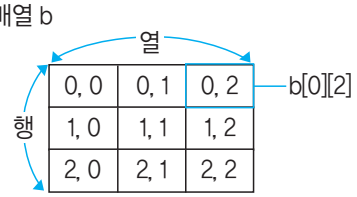
핵심 259 2차원 배열

- 2차원 배열은 변수들을 평면, 즉 행과 열로 조합한 배열이다.
- 형식

자료형 변수명[행개수][열개수]	• 자료형 : 배열에 저장할 자료의 형을 지정한다. • 변수명 : 사용할 배열의 이름으로 사용자가 임의로 지정한다. • 행개수 : 배열의 행 크기를 지정한다. • 열개수 : 배열의 열 크기를 지정한다.

예 int b[3][3] : 3개의 행과 3개의 열을 갖는 정수형 배열 b

배열 b



b[0][2] : b는 배열의 이름이고, 0은 행 첨자, 2는 열 첨자로서 배열 b에서의 위치를 나타낸다.

예제 3행 4열의 배열에 다음과 같이 숫자 저장하기

배열 a

1	2	3	4
5	6	7	8
9	10	11	12

```
#include <stdio.h>
main( )
{
  int a[3][4];
  int i, j, k = 0;
  for (i = 0; i < 3; i++) ❶

  { ❷
    for (j = 0; j < 4; j++) ❸



    { ❹
      k++; ❺
      a[i][j] = k; ❻
    } ❼
  } ❽
}
```

3행 4열의 크기를 갖는 정수형 배열 a를 선언한다.

정수형 변수 i, j, k를 선언하고 k를 0으로 초기화 한다.

반복 변수 i가 0에서 시작하여 1씩 증가하면서 3보다 작은 동안 ❷~❽번을 반복하여 수행한다. 결국 ❸번 문장을 3회 반복한다.

❷~❽이 ❶번 반복문의 반복 범위이지만 실제 실행할 문장은 ❸번 하나이다.

반복 변수 j가 0에서 시작하여 1씩 증가하면서 4보다 작은 동안 ❹~❼번을 반복하여 수행한다.
- i가 0일 때 j는 0에서 3까지 4회 반복
- i가 1일 때 j는 0에서 3까지 4회 반복
- i가 2일 때 j는 0에서 3까지 4회 반복 수행하므로 ❺~❻번을 총 12회 수행한다.

❹~❼이 ❸번 반복문의 반복 범위이다.

k를 1씩 증가시킨다. k는 총 12회 증가하므로 1~12까지 변한다.

배열 a의 i행 j열에 k를 기억시킨다. a[0][0]~a[2][3]까지 1~12가 저장된다.

❹번의 짝이다.

❷번의 짝이다.

첫 번째 { 의 짝이자 프로그램의 끝이다.

핵심 260 배열의 초기화

- 배열 선언 시 초기값을 지정할 수 있다.
- 배열을 선언할 때 배열의 크기를 생략하는 경우에는 반드시 초기값을 지정해야 초기값을 지정한 개수 만큼의 배열이 선언된다.

예 1차원 배열 초기화

방법 1 char a[3] = {'A', 'B', 'C'}

방법 2 char a[ ] = {'A', 'B', 'C'}

배열 a

A	B	C
a[0]	a[1]	a[2]

예 2차원 배열 초기화

방법 1 int a[2][4] = { {10, 20, 30, 40}, {50, 60, 70, 80} };

방법 2 int a[2][4] = {10, 20, 30, 40, 50, 60, 70, 80}

배열 a

a[0][0]	a[0][1]	a[0][2]	a[0][3]
10	20	30	40
50	60	70	80
a[1][0]	a[1][1]	a[1][2]	a[1][3]

- 배열의 개수보다 적은 수로 배열을 초기화하면 입력된 값만큼 지정한 숫자가 입력되고, 나머지 요소에는 0이 입력된다.

예 int a[5] = { 3, }; 또는 int a[5] = { 3 };

배열 a

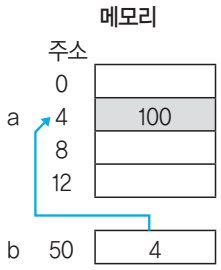
3	0	0	0	0
a[0]	a[1]	a[2]	a[3]	a[4]

핵심 261 포인터와 포인터 변수

포인터는 변수의 주소를 말하며, C언어에서는 주소를 제어할 수 있는 기능을 제공한다.

- C언어에서 변수의 주소를 저장할 때 사용하는 변수를 포인터 변수라 한다.
- 포인터 변수를 선언할 때는 자료의 형을 먼저 쓰고 변수명 앞에 간접 연산자 *를 붙인다(예 int *a;).
- 포인터 변수에 주소를 저장하기 위해 변수의 주소를 알아낼 때는 변수 앞에 번지 연산자 &를 붙인다(예 a = &b;).
- 실행문에서 포인터 변수에 간접 연산자 *를 붙이면 해당 포인터 변수가 가리키는 곳의 값을 말한다(예 c = *a;).
- 포인터 변수는 필요에 의해 동적으로 할당되는 메모리 영역인 힙 영역에 접근하는 동적 변수이다.

예를 들어, a 변수에 100을 저장시키고, a 변수의 주소를 포인터 변수 b에 기억시켰다면 다음 그림과 같이 표현하고 말할 수 있다.



- a는 메모리의 4번지에 대한 이름이다.
- a 변수의 주소는 4다.
- a 변수에는 100이 기억되어 있다.
- 4번지에는 100이 기억되어 있다.
- &a는 a 변수의 주소를 말한다. 즉 &a는 4다.
- 포인터 변수 b는 a 변수의 주소를 기억하고 있다.
- 포인터 변수가 가리키는 곳의 값을 말할 때는 *을 붙인다.
- *b는 b에 저장된 주소가 가리키는 곳에 저장된 값을 말하므로 100이다.

예제 1 다음 C언어로 구현된 프로그램의 출력 결과를 확인하시오.

```
main( )
{
  int a = 50; ❶
  int *b; ❷
  b = &a; ❸
  *b = *b+20; ❹

  printf("%d, %d", a, *b); ❺
}
```

❶ 정수형 변수 a를 선언하고 50으로 초기화한다.

❷ 정수형 변수가 저장된 곳의 주소를 기억할 포인터 변수 b를 선언한다.

❸ 정수형 변수 a의 주소를 포인터 변수 b에 기억시킨다. b에는 a의 주소가 저장된다.

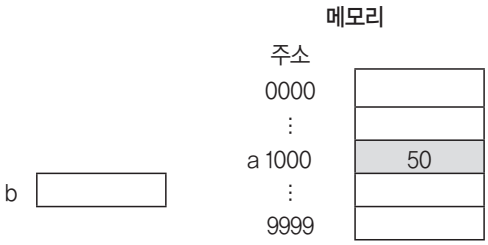
❹ b가 가리키는 곳의 값에 20을 더한다. b가 가리키는 곳이 a이므로 결국 a의 값도 바뀌는 것이다.

❺ 결과 70, 70

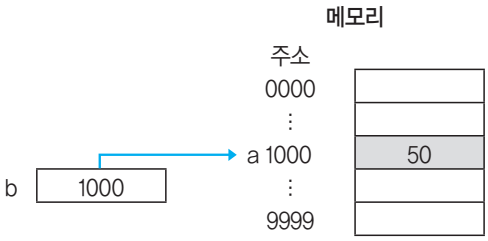
- ❷와 같이 선언할 때 *는 해당 변수가 포인터 변수라는 것을 의미한다.
- ❹, ❺와 같이 사용할 때 *를 붙이면 그 포인터 변수가 가리키는 곳의 값을 의미한다.

위 코드의 실행 과정에 따라 메모리의 변화를 그려보면 다음과 같다.

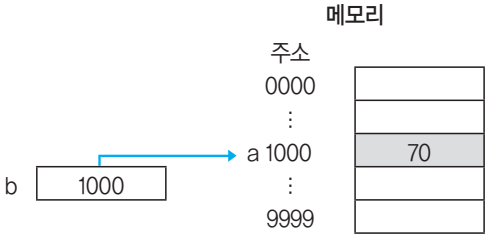
❶, ❷번 수행 : 주기억장치의 빈 공간 어딘가에 a라는 이름을 붙이고 그 곳에 50을 저장한다.



❸번 수행 : 변수 a의 주소가 b에 기억된다는 것은 b가 변수 a의 주소를 가리키고 있다는 의미이다.



❹번 수행 : b가 가리키는 곳의 값에 20을 더해 다시 b가 가리키는 곳에 저장한다. 그곳은 변수 a의 주소이므로 변수 a의 값도 저절로 변경되는 것이다.



핵심 262 Python의 기초

- 변수의 자료형에 대한 선언이 없다.
- 문장의 끝을 의미하는 세미콜론(;)을 사용할 필요가 없다.
- 변수에 연속하여 값을 저장하는 것이 가능하다.

  예 x, y, z = 10, 20, 30
- if나 for와 같이 코드 블록을 포함하는 명령문을 작성할 때 코드 블록은 콜론(:)과 여백으로 구분한다.
- 여백은 일반적으로 4칸 또는 한 개의 탭만큼 띄워야하며, 같은 수준의 코드들은 반드시 동일한 여백을 가져야 한다.

input( ) 함수

- input( ) 함수는 Python의 표준 입력 함수로, 키보드로 입력받아 변수에 저장하는 함수이다.

• 형식

변수 = input(출력문자)

• '출력문자'는 생략이 가능하며, '변수'는 사용자가 임의로 지정할 수 있다.
• 값을 입력하고 Enter를 누르면, 입력한 값이 '변수'에 저장된다.

예 a = input('입력하세요.') → 화면에 입력하세요.가 출력되고 그 뒤에서 커서가 깜빡거리며 입력을 기다린다. 키보드로 값을 입력하면 변수 a에 저장된다.

print( ) 함수

• 형식1

print(출력값1, 출력값2, …, sep = 분리문자, end = 종료문자)

• '출력값'에는 숫자, 문자, 문자열, 변수 등 다양한 값이나 식이 올 수 있다.
• 'sep'는 여러 값을 출력할 때 값과 값 사이를 구분하기 위해 출력하는 문자로, 생략할 경우 기본값은 공백 한 칸(' ')이다.
• 'end'는 맨 마지막에 표시할 문자로, 생략할 경우 기본값은 줄 나눔이다.

예 print(82, 24, sep = '–', end = ',') → 82와 24 사이에 분리문자 '–'가 출력되고, 마지막에 종료문자 ','가 출력된다.

결과 82–24,

슬라이스(Slice)

슬라이스는 문자열이나 리스트와 같은 순차형 객체에서 일부를 잘라(slicing) 반환하는 기능이다.

• 형식

객체명[초기위치:최종위치]
'초기위치'에서 '최종위치'–1까지의 요소들을 가져온다.

객체명[초기위치:최종위치:증가값]
• '초기위치'에서 '최종위치'–1까지 '증가값'만큼 증가하면서 해당 위치의 요소들을 가져온다.
• '증가값'이 음수인 경우 '초기위치'에서 '최종위치'+1까지 '증가값' 만큼 감소하면서 해당 위치의 요소들을 가져온다.

• 슬라이스는 일부 인수를 생략하여 사용할 수 있다.

객체명[:] 또는 객체명[::]
객체의 모든 요소를 반환한다.

객체명[초기위치:]
객체의 '초기위치'에서 마지막 위치까지의 요소들을 반환한다.

객체명[:최종위치]
객체의 0번째 위치에서 '최종위치'–1까지의 요소들을 반환한다.

객체명[::증가값]
객체의 0번째 위치에서 마지막 위치까지 '증가값'만큼 증가하면서 해당 위치의 요소들을 반환한다.

예 a = ['a', 'b', 'c', 'd', 'e']일 때
a[1:3] → ['b', 'c']
a[0:5:2] → ['a', 'c', 'e']
a[3:] → ['d', 'e']
a[:3] → ['a', 'b', 'c']
a[::3] → ['a', 'd']

핵심 263 절차적 프로그래밍 언어

절차적 프로그래밍 언어의 개요

절차적 프로그래밍 언어는 일련의 처리 절차를 정해진 문법에 따라 순서대로 기술해 나가는 언어이다.

- 절차적 프로그래밍 언어는 프로그램이 실행되는 절차를 중요시 한다.
- 절차적 프로그래밍 언어는 데이터를 중심으로 프로시저를 구현하며, 프로그램 전체가 유기적으로 연결되어 있다.
- 절차적 프로그래밍 언어는 자연어에 가까운 단어와 문장으로 구성된다.
- 절차적 프로그래밍 언어는 과학 계산이나 하드웨어 제어에 주로 사용된다.

절차적 프로그래밍 언어의 장 · 단점

- 컴퓨터의 처리 구조와 유사하여 실행 속도가 빠르다.
- 같은 코드를 복사하지 않고 다른 위치에서 호출하여 사용할 수 있다.
- 모듈 구성이 용이하며, 구조적인 프로그래밍이 가능하다.
- 프로그램을 분석하기 어렵다.
- 유지 보수나 코드의 수정이 어렵다.

핵심 264 절차적 프로그래밍 언어의 종류

언어	특징
C	• 1972년 미국 벨 연구소의 데니스 리치에 의해 개발되었다. • 시스템 소프트웨어를 개발하기 편리하여 시스템 프로그래밍 언어로 널리 사용된다. • 자료의 주소를 조작할 수 있는 포인터를 제공한다. • 고급 프로그래밍 언어이면서 저급 프로그램 언어의 특징을 모두 갖췄다. • UNIX의 일부가 C 언어로 구현되었다. • 컴파일러 방식의 언어이다. • 이식성이 좋아 컴퓨터 기종에 관계없이 프로그램을 작성할 수 있다.
ALGOL	• 수치 계산이나 논리 연산을 위한 과학 기술 계산용 언어이다. • PASCAL과 C 언어의 모체가 되었다.
COBOL	• 사무 처리용 언어이다. • 영어 문장 형식으로 구성되어 있어 이해와 사용이 쉽다. • 4개의 DIVISION으로 구성되어 있다.
FORTRAN	• 과학 기술 계산용 언어이다. • 수학과 공학 분야의 공식이나 수식과 같은 형태로 프로그래밍 할 수 있다.

핵심 265 객체지향 프로그래밍 언어

객체지향 프로그래밍 언어의 개요

객체지향 프로그래밍 언어는 현실 세계의 개체(Entity)를 기계의 부품처럼 하나의 객체로 만들어, 기계적인 부품들을 조립하여 제품을 만들 듯이 소프트웨어를 개발할 때도 객체들을 조립해서 프로그램을 작성할 수 있도록 한 프로그래밍 기법이다.

- 프로시저보다는 명령과 데이터로 구성된 객체를 중심으로 하는 프로그래밍 기법으로, 한 프로그램을 다른 프로그램에서 이용할 수 있도록 한다.

객체지향 프로그래밍 언어의 장 · 단점

- 상속을 통한 재사용과 시스템의 확장이 용이하다.
- 코드의 재활용성이 높다.
- 자연적인 모델링에 의해 분석과 설계를 쉽고 효율적으로 할 수 있다.
- 사용자와 개발자 사이의 이해를 쉽게 해준다.
- 대형 프로그램의 작성이 용이하다.
- 소프트웨어 개발 및 유지보수가 용이하다.
- 프로그래밍 구현을 지원해 주는 정형화된 분석 및 설계 방법이 없다.
- 구현 시 처리 시간이 지연된다.

객체지향 프로그래밍 언어의 종류

언어	특징
JAVA	• 분산 네트워크 환경에 적용이 가능하며, 멀티스레드 기능을 제공하므로 여러 작업을 동시에 처리할 수 있다. • 운영체제 및 하드웨어에 독립적이며, 이식성이 강하다. • 캡슐화가 가능하고 재사용성 높다.
C++	• C 언어에 객체지향 개념을 적용한 언어이다. • 모든 문제를 객체로 모델링하여 표현한다.
Smalltalk	• 1세대 객체지향 프로그래밍 언어 중 하나로 순수한 객체지향 프로그래밍 언어이다. • 최초로 GUI를 제공한 언어이다.

핵심 266 스크립트 언어

스크립트 언어(Script Language)의 개요

스크립트 언어는 HTML 문서 안에 직접 프로그래밍 언어를 삽입하여 사용하는 것으로, 기계어로 컴파일 되지 않고 별도의 번역기가 소스를 분석하여 동작하게 하는 언어이다.

- 게시판 입력, 상품 검색, 회원 가입 등과 같은 데이터베이스 처리 작업을 수행하기 위해 주로 사용한다.
- 스크립트 언어는 클라이언트의 웹 브라우저에서 해석되어 실행되는 클라이언트용 스크립트 언어와 서버에서 해석되어 실행된 후 결과만 클라이언트로 보내는 서버용 스크립트 언어가 있다.
  - 서버용 스크립트 언어 : ASP, JSP, PHP, 파이썬
  - 클라이언트용 스크립트 언어 : 자바 스크립트(JAVA Script), VB 스크립트(Visual Basic Script)

스크립트 언어의 장 · 단점

- 컴파일 없이 바로 실행하므로 결과를 바로 확인할 수 있다.
- 배우고 코딩하기 쉽다.
- 개발 시간이 짧다.
- 소스 코드를 쉽고 빠르게 수정할 수 있다.
- 코드를 읽고 해석해야 하므로 실행 속도가 느리다.
- 런타임 오류가 많이 발생한다.

스크립트 언어의 종류

자바 스크립트 (JAVA Script)	• 웹 페이지의 동작을 제어하는 데 사용되는 클라이언트용 스크립트 언어로, 클래스가 존재하지 않으며 변수 선언도 필요 없다. • 서버에서 데이터를 전송할 때 아이디, 비밀번호, 수량 등의 입력 사항을 확인하기 위한 용도로 많이 사용된다.
VB 스크립트 (Visual Basic Script)	마이크로소프트 사에서 자바 스크립트에 대응하기 위해 제작한 언어로, Active X를 사용하여 마이크로소프트 사의 애플리케이션들을 컨트롤할 수 있다.
ASP(Active Server Page)	• 서버 측에서 동적으로 수행되는 페이지를 만들기 위한 언어로 마이크로 소프트 사에서 제작하였다. • Windows 계열에서만 수행 가능한 프로그래밍 언어이다.
JSP(Java Server Page)	JAVA로 만들어진 서버용 스크립트로, 다양한 운영체제에서 사용이 가능하다.
PHP (Professional Hypertext Preprocessor)	• 서버용 스크립트 언어로, Linux, Unix, Windows 운영체제에서 사용 가능하다. • C, Java 등과 문법이 유사하므로 배우기 쉬워 웹 페이지 제작에 많이 사용된다.
파이썬(Python)	객체지향 기능을 지원하는 대화형 인터프리터 언어로, 플랫폼에 독립적이고 문법이 간단하여 배우기 쉽다.
쉘 스크립트	• 유닉스/리눅스 계열의 쉘(Shell)에서 사용되는 명령어들의 조합으로 구성된 스크립트 언어이다. • 컴파일 단계가 없어 실행 속도가 빠르다. • 저장 시 확장자로 '.sh'가 붙는다. • 쉘의 종류 : Bash Shell, Bourne Shell, C Shell, Korn Shell 등 • 쉘 스크립트에서 사용되는 제어문 – 선택형 : if, case – 반복형 : for, while, until
Basic	절차지향 기능을 지원하는 대화형 인터프리터 언어로, 초보자도 쉽게 사용할 수 있는 문법 구조를 갖는다.

핵심 267 선언형 언어

선언형 언어

선언형 언어는 명령형 언어와 반대되는 개념의 언어로, 명령형 언어가 문제를 해결하기 위한 방법을 기술한다면 선언형 언어는 프로그램이 수행해야 하는 문제를 기술하는 언어이다.

- 선언형 언어는 목표를 명시하고 알고리즘은 명시하지 않는다.
- 선언형 언어에는 함수형 언어와 논리형 언어 등이 있다.

함수형 언어	• 수학적 함수를 조합하여 문제를 해결하는 언어로, 알려진 값을 함수에 적용하는 것을 기반으로 한다. • 적용형 언어라고도 한다. • 종류 : LISP
논리형 언어	• 기호 논리학에 기반을 둔 언어로, 논리 문장을 이용하여 프로그램을 표현하고 계산을 수행한다. • 선언적 언어라고도 한다. • 종류 : PROLOG

※ 명령형 언어

명령형 언어는 순차적인 명령 수행을 기본으로 하는 언어로, 문제를 처리하기 위한 방법에 초점을 두고 코드를 작성한다.

- 폰노이만 구조에 개념적인 기초를 두고 있다.
- 명령형 언어는 알고리즘을 명시하고 목표는 명시하지 않는다.
- 명령형 언어에는 절차적 언어와 객체지향 언어가 있다.
- 종류 : FORTRAN, COBOL, C, JAVA 등

선언형 프로그래밍 언어 종류

HTML	인터넷의 표준 문서인 하이퍼텍스트 문서를 만들기 위해 사용하는 언어로, 특별한 데이터 타입이 없는 단순한 텍스트이므로 호환성이 좋고 사용이 편리하다.
LISP	• 인공지능 분야에 사용되는 언어이다. • 기본 자료 구조가 연결 리스트 구조이며, 재귀(Recursion) 호출을 많이 사용한다.
PROLOG	논리학을 기초로 한 고급 언어로, 인공 지능 분야에서의 논리적인 추론이나 리스트 처리 등에 주로 사용된다.
XML	• 기존 HTML의 단점을 보완하여 웹에서 구조화된 폭넓고 다양한 문서들을 상호 교환할 수 있도록 설계된 언어이다. • HTML에 사용자가 새로운 태그(Tag)를 정의할 수 있으며, 문서의 내용과 이를 표현하는 방식이 독립적이다.
Haskell	• 함수형 프로그래밍 언어로 부작용(Side Effect)이 없다. • 코드가 간결하고 에러 발생 가능성이 낮다.

핵심 268 라이브러리

라이브러리의 개념

라이브러리는 프로그램을 효율적으로 개발할 수 있도록 자주 사용하는 함수나 데이터들을 미리 만들어 모아 놓은 집합체이다.

- 자주 사용하는 함수들의 반복적인 코드 작성을 피하기 위해 미리 만들어 놓은 것으로, 필요할 때는 언제든지 호출하여 사용할 수 있다.
- 라이브러리에는 표준 라이브러리와 외부 라이브러리가 있다.
  - 표준 라이브러리 : 프로그래밍 언어에 기본적으로 포함되어 있는 라이브러리로, 여러 종류의 모듈이나 패키지 형태이다.
  - 외부 라이브러리 : 개발자들이 필요한 기능들을 만들어 인터넷 등에 공유해 놓은 것으로, 외부 라이브러리를 다운받아 설치한 후 사용한다.

C언어의 대표적인 표준 라이브러리

C언어는 라이브러리를 헤더 파일로 제공하는데, 각 헤더 파일에는 응용 프로그램 개발에 필요한 함수들이 정리되어 있다.

- C언어에서 헤더 파일을 사용하려면 '#include 〈stdio.h〉'와 같이 include문을 이용해 선언한 후 사용해야 한다.
- 종류 : stdio.h, math.h, string.h, stdlib.h, time.h

JAVA의 대표적인 표준 라이브러리

JAVA는 라이브러리를 패키지에 포함하여 제공하는데, 각 패키지에는 JAVA 응용 프로그램 개발에 필요한 메소드들이 클래스로 정리되어 있다.

- JAVA에서 패키지를 사용하려면 'import java.util'과 같이 import문을 이용해 선언한 후 사용해야 한다.
- 종류 : java.lang, java.util, java.io, java.net, java.awt

핵심 269 예외 처리

프로그램의 정상적인 실행을 방해하는 조건이나 상태를 예외(Exception)라고 하며, 이러한 예외가 발생했을 때 프로그래머가 해당 문제에 대비해 작성해 놓은 처리 루틴을 수행하도록 하는 것을 예외 처리(Exception Handling)라고 한다.

- 예외가 발생했을 때 일반적인 처리 루틴은 프로그램을 종료시키거나 로그를 남기도록 하는 것이다.
- C++, Ada, JAVA, 자바스크립트와 같은 언어에는 예외 처리 기능이 내장되어 있으며, 그 외의 언어에서는 필요한 경우 조건문을 이용해 예외 처리 루틴을 작성한다.
- 예외의 원인에는 컴퓨터 하드웨어 문제, 운영체제의 설정 실수, 라이브러리 손상, 사용자의 입력 실수, 받아들일 수 없는 연산, 할당하지 못하는 기억장치 접근 등 다양하다.

핵심 270 프로토타입(Prototype)

프로그래밍 언어에서 프로토타입이란 함수 원형(Function Prototype)이라는 의미로, 컴파일러에게 사용될 함수에 대한 정보를 미리 알리는 것이다.

- 함수가 호출되기 전에 함수가 미리 정의되는 경우에는 프로토타입을 정의하지 않아도 된다.
- 프로토타입은 본문이 없다는 점을 제외하고 함수 정의와 형태가 동일하다.
- 프로토타입에 정의된 반환 형식은 함수 정의에 지정된 반환 형식과 반드시 일치해야 한다.

핵심 271 운영체제의 정의 및 목적

운영체제의 정의

운영체제(OS; Operating System)는 컴퓨터 시스템의 자원들을 효율적으로 관리하며, 사용자가 컴퓨터를 편리하고 효과적으로 사용할 수 있도록 환경을 제공하는 여러 프로그램의 모임이다.

- 컴퓨터 사용자와 컴퓨터 하드웨어 간의 인터페이스로서 동작하는 시스템 소프트웨어의 일종으로, 다른 응용 프로그램이 유용한 작업을 할 수 있도록 환경을 제공해준다.

운영체제의 목적

운영체제의 목적에는 처리 능력 향상, 사용 가능도 향상, 신뢰도 향상, 반환 시간 단축 등이 있다.

- 처리 능력, 반환 시간, 사용 가능도, 신뢰도는 운영체제의 성능을 평가하는 기준이 된다.

처리 능력 (Throughput)	일정 시간 내에 시스템이 처리하는 일의 양
반환 시간 (Turn Around Time)	시스템에 작업을 의뢰한 시간부터 처리가 완료될 때까지 걸린 시간
사용 가능도 (Availability)	시스템을 사용할 필요가 있을 때 즉시 사용 가능한 정도
신뢰도(Reliability)	시스템이 주어진 문제를 정확하게 해결하는 정도

핵심 272 운영체제의 기능

- 프로세서(처리기, Processor), 기억장치(주기억장치, 보조기억장치), 입 · 출력장치, 파일 및 정보 등의 자원을 관리한다.
- 자원을 효율적으로 관리하기 위해 자원의 스케줄링 기능을 제공한다.
- 사용자와 시스템 간의 편리한 인터페이스를 제공한다.
- 시스템의 각종 하드웨어와 네트워크를 관리 · 제어한다.
- 데이터를 관리하고, 데이터 및 자원의 공유 기능을 제공한다.
- 시스템의 오류를 검사하고 복구한다.
- 자원 보호 기능을 제공한다.
- 입 · 출력에 대한 보조 기능을 제공한다.
- 가상 계산기 기능을 제공한다.

핵심 273 Windows

- Windows는 1990년대 마이크로소프트(Microsoft) 사가 개발한 운영체제이다.
- Windows의 주요 특징

그래픽 사용자 인터페이스 (GUI; Graphic User Interface)	키보드로 명령어를 직접 입력하지 않고, 마우스로 아이콘이나 메뉴를 선택하여 모든 작업을 수행하는 방식을 말한다.
선점형 멀티태스킹 (Preemptive Multi-Tasking)	동시에 여러 개의 프로그램을 실행하는 멀티태스킹을 하면서 운영체제가 각 작업의 CPU 이용 시간을 제어하여 응용 프로그램 실행중 문제가 발생하면 해당 프로그램을 강제 종료시키고 모든 시스템 자원을 반환하는 방식을 말한다.
PnP (Plug and Play, 자동 감지 기능)	컴퓨터 시스템에 프린터나 사운드 카드 등의 하드웨어를 설치했을 때, 해당 하드웨어를 사용하는데 필요한 시스템 환경을 운영체제가 자동으로 구성해 주는 기능이다.
OLE (Object Linking and Embedding)	다른 여러 응용 프로그램에서 작성된 문자나 그림 등의 개체(Object)를 현재 작성 중인 문서에 자유롭게 연결(Linking)하거나 삽입(Embedding)하여 편집할 수 있게 하는 기능이다.
255자의 긴 파일명	• Windows에서는 파일 이름을 지정할 때 VFAT(Virtual File Allocation Table)를 이용하여 최대 255자까지 지정할 수 있다. • 파일 이름으로는 ₩ / : * ? " < > | 를 제외한 모든 문자 및 공백을 사용할 수 있으며, 한글의 경우 127자까지 지정할 수 있다.
Single-User 시스템	컴퓨터 한 대를 한 사람만이 독점해서 사용한다.

핵심 274 UNIX의 개요 및 특징

UNIX는 1960년대 AT&T 벨(Bell) 연구소, MIT, General Electric이 공동 개발한 운영체제이다.

- 시분할 시스템(Time Sharing System)을 위해 설계된 대화식 운영체제로, 소스가 공개된 개방형 시스템(Open System)이다.
- 대부분 C 언어로 작성되어 있어 이식성이 높으며 장치, 프로세스 간의 호환성이 높다.
- 크기가 작고 이해하기가 쉽다.
- 다중 사용자(Multi-User), 다중 작업(Multi-Tasking)을 지원한다.
- 많은 네트워킹 기능을 제공하므로 통신망(Network) 관리용 운영체제로 적합하다.
- 트리 구조의 파일 시스템을 갖는다.
- 전문적인 프로그램 개발에 용이하다.
- 다양한 유틸리티 프로그램들이 존재한다.

※ 다중 사용자(Multi-User), 다중 작업(Multi-Tasking)

- 다중 사용자(Multi-User)는 여러 사용자가 동시에 시스템을 사용하는 것이고, 다중 작업(Multi-Tasking)은 여러 개의 작업이나 프로그램을 동시에 수행하는 것을 의미한다.
- 하나 이상의 작업을 백그라운드에서 수행하므로 여러 작업을 동시에 처리할 수 있다.

핵심 275 UNIX 시스템의 구성

커널(Kernel)

- UNIX의 가장 핵심적인 부분이다.
- 컴퓨터가 부팅될 때 주기억장치에 적재된 후 상주하면서 실행된다.
- 하드웨어를 보호하고, 프로그램과 하드웨어 간의 인터페이스 역할을 담당한다.
- 프로세스(CPU 스케줄링) 관리, 기억장치 관리, 파일 관리, 입 · 출력 관리, 프로세스간 통신, 데이터 전송 및 변환 등 여러 가지 기능을 수행한다.

쉘(Shell)

- 사용자의 명령어를 인식하여 프로그램을 호출하고 명령을 수행하는 명령어 해석기이다.
- 시스템과 사용자 간의 인터페이스를 담당한다.
- DOS의 COMMAND.COM과 같은 기능을 수행한다.
- 주기억장치에 상주하지 않고, 명령어가 포함된 파일 형태로 존재하며 보조 기억장치에서 교체 처리가 가능하다.
- 파이프라인 기능을 지원하고 입 · 출력 재지정을 통해 출력과 입력의 방향을 변경할 수 있다.
- 공용 Shell(Bourne Shell, C Shell, Korn Shell)이나 사용자 자신이 만든 Shell을 사용할 수 있다.

Utility Program

- 일반 사용자가 작성한 응용 프로그램을 처리하는 데 사용한다.
- DOS에서의 외부 명령어에 해당된다.
- 유틸리티 프로그램에는 에디터, 컴파일러, 인터프리터, 디버거 등이 있다.

핵심 276 기억장치 관리 전략

기억장치의 관리 전략은 보조기억장치의 프로그램이나 데이터를 주기억장치에 적재시키는 시기, 적재 위치 등을 지정하여 한정된 주기억장치의 공간을 효율적으로 사용하기 위한 것으로 반입(Fetch) 전략, 배치(Placement) 전략, 교체(Replacement) 전략이 있다.

반입(Fetch) 전략

반입 전략은 보조기억장치에 보관중인 프로그램이나 데이터를 언제 주기억장치로 적재할 것인지를 결정하는 전략이다.

요구 반입 (Demand Fetch)	실행중인 프로그램이 특정 프로그램이나 데이터 등의 참조를 요구할 때 적재하는 방법이다.
예상 반입 (Anticipatory Fetch)	실행중인 프로그램에 의해 참조될 프로그램이나 데이터를 미리 예상하여 적재하는 방법이다.

배치(Placement) 전략

배치 전략은 새로 반입되는 프로그램이나 데이터를 주기억장치의 어디에 위치시킬 것인지를 결정하는 전략이다.

최초 적합 (First Fit)	프로그램이나 데이터가 들어갈 수 있는 크기의 빈 영역 중에서 첫 번째 분할 영역에 배치시키는 방법이다.
최적 적합 (Best Fit)	프로그램이나 데이터가 들어갈 수 있는 크기의 빈 영역 중에서 단편화를 가장 작게 남기는 분할 영역에 배치시키는 방법이다.
최악 적합 (Worst Fit)	프로그램이나 데이터가 들어갈 수 있는 크기의 빈 영역 중에서 단편화를 가장 많이 남기는 분할 영역에 배치시키는 방법이다.

교체(Replacement) 전략

교체 전략은 주기억장치의 모든 영역이 이미 사용중인 상태에서 새로운 프로그램이나 데이터를 주기억장치에 배치하려고 할 때, 이미 사용되고 있는 영역 중에서 어느 영역을 교체하여 사용할 것인지를 결정하는 전략이다.

- 교체 전략에는 FIFO, OPT, LRU, LFU, NUR, SCR 등이 있다.

핵심 277 주기억장치 할당의 개념

주기억장치 할당 기법은 프로그램이나 데이터를 실행시키기 위해 주기억장치에 어떻게 할당할 것인지에 대한 내용이며, 연속 할당 기법과 분산 할당 기법으로 분류할 수 있다.

연속 할당 기법	프로그램을 주기억장치에 연속으로 할당하는 기법으로, 단일 분할 할당 기법과 다중 분할 할당 기법이 있다. • 단일 분할 할당 기법 : 오버레이, 스와핑 • 다중 분할 할당 기법 : 고정 분할 할당 기법, 동적 분할 할당 기법
분산 할당 기법	프로그램을 특정 단위의 조각으로 나누어 주기억장치 내에 분산하여 할당하는 기법으로, 페이징 기법과 세그먼테이션 기법으로 나눌 수 있다.

핵심 278 다중 분할 할당 기법

고정 분할 할당(Multiple contiguous Fixed parTition allocation, MFT) 기법 = 정적 할당(Static Allocation) 기법

고정 분할 할당은 프로그램을 할당하기 전에 운영체제가 주기억장치의 사용자 영역을 여러 개의 고정된 크기로 분할하고 준비상태 큐에서 준비중인 프로그램을 각 영역에 할당하여 수행하는 기법이다.

- 프로그램을 실행하려면 프로그램 전체가 주기억장치에 위치해야 한다.
- 프로그램이 분할된 영역보다 커서 영역 안에 들어갈 수 없는 경우가 발생할 수 있다.
- 일정한 크기의 분할 영역에 다양한 크기의 프로그램이 할당되므로 내부 단편화 및 외부 단편화가 발생하여 주기억장치의 낭비가 많다.
- 실행할 프로그램의 크기를 미리 알고 있어야 한다.
- 다중 프로그래밍을 위해 사용되었으나 현재는 사용되지 않는다.

가변 분할 할당(Multiple contiguous Variable parTition allocation, MVT) 기법 = 동적 할당(Dynamic Allocation) 기법

고정 분할 할당 기법의 단편화를 줄이기 위한 것으로, 미리 주기억장치를 분할해 놓는 것이 아니라 프로그램을 주기억장치에 적재하면서 필요한 만큼의 크기로 영역을 분할하는 기법이다.

- 주기억장치를 효율적으로 사용할 수 있으며, 다중 프로그래밍의 정도를 높일 수 있다.
- 고정 분할 할당 기법에 비해 실행될 프로세스 크기에 대한 제약이 적다.
- 단편화를 상당 부분 해결할 수 있으나 영역과 영역 사이에 단편화가 발생될 수 있다.

핵심 279 가상기억장치

가상기억장치는 보조기억장치(하드디스크)의 일부를 주기억장치처럼 사용하는 것으로, 용량이 작은 주기억장치를 마치 큰 용량을 가진 것처럼 사용하는 기법이다.

- 주기억장치의 용량보다 큰 프로그램을 실행하기 위해 사용한다.
- 주기억장치의 이용률과 다중 프로그래밍의 효율을 높일 수 있다.
- 가상기억장치에 저장된 프로그램을 실행하려면 가상기억장치의 주소를 주기억장치의 주소로 바꾸는 주소 변환 작업이 필요하다.
- 블록 단위로 나누어 사용하므로 연속 할당 방식에서 발생할 수 있는 단편화를 해결할 수 있다.
- 가상기억장치의 일반적인 구현 방법에는 블록의 종류에 따라 페이징 기법과 세그먼테이션 기법으로 나눌 수 있다.

페이징(Paging) 기법

- 페이징 기법은 가상기억장치에 보관되어 있는 프로그램과 주기억장치의 영역을 동일한 크기로 나눈 후 나눠진 프로그램(페이지)을 동일하게 나눠진 주기억장치의 영역(페이지 프레임)에 적재시켜 실행하는 기법이다.

- 프로그램을 일정한 크기로 나눈 단위를 페이지(Page)라고 하고, 페이지 크기로 일정하게 나누어진 주기억장치의 단위를 페이지 프레임(Page Frame)이라고 한다.
- 외부 단편화는 발생하지 않으나 내부 단편화는 발생할 수 있다.
- 주소 변환을 위해서 페이지의 위치 정보를 가지고 있는 페이지 맵 테이블(Page Map Table)이 필요하다.

세그먼테이션(Segmentation) 기법

- 세그먼테이션 기법은 가상기억장치에 보관되어 있는 프로그램을 다양한 크기의 논리적인 단위로 나눈 후 주기억장치에 적재시켜 실행시키는 기법이다.
- 프로그램을 배열이나 함수 등과 같은 논리적인 크기로 나눈 단위를 세그먼트(Segment)라고 하며, 각 세그먼트는 고유한 이름과 크기를 갖는다.
- 주소 변환을 위해서 세그먼트가 존재하는 위치 정보를 가지고 있는 세그먼트 맵 테이블(Segment Map Table)이 필요하다.
- 내부 단편화는 발생하지 않으나 외부 단편화는 발생할 수 있다.

핵심 280 페이지 교체 알고리즘

페이지 교체 알고리즘은 페이지 부재(Page Fault)가 발생했을 때 가상기억장치의 필요한 페이지를 주기억장치에 적재해야 하는데, 이때 주기억장치의 모든 페이지 프레임이 사용중이면 어떤 페이지 프레임을 선택하여 교체할 것인지를 결정하는 기법이다.

OPT (OPTimal replacement, 최적 교체)	• 앞으로 가장 오랫동안 사용하지 않을 페이지를 교체하는 기법이다. • 벨레이디(Belady)가 제안한 것으로, 페이지 부재 횟수가 가장 적게 발생하는 가장 효율적인 알고리즘이다.
FIFO (First In First Out)	• 각 페이지가 주기억장치에 적재될 때마다 그때의 시간을 기억시켜 가장 먼저 들어와서 가장 오래 있었던 페이지를 교체하는 기법이다. • 이해하기 쉽고, 프로그래밍 및 설계가 간단하다.
LRU (Least Recently Used)	• 최근에 가장 오랫동안 사용하지 않은 페이지를 교체하는 기법이다. • 각 페이지마다 계수기(Counter)나 스택(Stack)을 두어 현 시점에서 가장 오랫동안 사용하지 않은, 즉 가장 오래 전에 사용된 페이지를 교체한다.
LFU (Least Frequently Used)	• 사용 빈도가 가장 적은 페이지를 교체하는 기법이다. • 활발하게 사용되는 페이지는 사용 횟수가 많아 교체되지 않고 사용된다.
SCR (Second Chance Replacement, 2차 기회 교체)	가장 오랫동안 주기억장치에 있던 페이지 중 자주 사용되는 페이지의 교체를 방지하기 위한 것으로, FIFO 기법의 단점을 보완하는 기법이다.
NUR (Not Used Recently)	• LRU와 비슷한 알고리즘으로, 최근에 사용하지 않은 페이지를 교체하는 기법이다. • 최근에 사용되지 않은 페이지는 향후에도 사용되지 않을 가능성이 높다는 것을 전제로, LRU에서 나타나는 시간적인 오버헤드를 줄일 수 있다. • 최근의 사용 여부를 확인하기 위해서 각 페이지마다 두 개의 비트, 즉 참조 비트(Reference Bit)와 변형 비트(Modified Bit, Dirty Bit)가 사용된다.

핵심 281 페이지 크기

페이지 크기가 작을 경우

- 페이지 단편화가 감소되고, 한 개의 페이지를 주기억장치로 이동하는 시간이 줄어든다.
- 불필요한 내용이 주기억장치에 적재될 확률이 적으므로 효율적인 워킹 셋을 유지할 수 있다.
- Locality에 더 일치할 수 있기 때문에 기억장치 효율이 높아진다.
- 페이지 정보를 갖는 페이지 맵 테이블의 크기가 커지고, 매핑 속도가 늦어진다.
- 디스크 접근 횟수가 많아져서 전체적인 입 · 출력 시간은 늘어난다.

페이지 크기가 클 경우

- 페이지 정보를 갖는 페이지 맵 테이블의 크기가 작아지고, 매핑 속도가 빨라진다.
- 디스크 접근 횟수가 줄어들어 전체적인 입 · 출력의 효율성이 증가된다.
- 페이지 단편화가 증가되고, 한 개의 페이지를 주기억장치로 이동하는 시간이 늘어난다.
- 프로세스(프로그램) 수행에 불필요한 내용까지도 주기억장치에 적재될 수 있다.

핵심 282 Locality

Locality(국부성, 지역성, 구역성, 국소성)는 프로세스가 실행되는 동안 주기억장치를 참조할 때 일부 페이지만 집중적으로 참조하는 성질이 있다는 이론이다.

- 스래싱을 방지하기 위한 워킹 셋 이론의 기반이 되었다.
- Locality의 종류에는 시간 구역성(Temporal Locality)과 공간 구역성(Spatial Locality)이 있다.

시간 구역성(Temporal Locality)

- 시간 구역성은 프로세스가 실행되면서 하나의 페이지를 일정 시간 동안 집중적으로 액세스하는 현상이다.
- 한 번 참조한 페이지는 가까운 시간 내에 계속 참조할 가능성이 높음을 의미한다.
- **시간 구역성이 이루어지는 기억 장소** : Loop(반복, 순환), 스택(Stack), 부 프로그램(Sub Routine), Counting(1씩 증감), 집계(Totaling)에 사용되는 변수(기억장소)

공간 구역성(Spatial Locality)

- 공간 구역성은 프로세스 실행 시 일정 위치의 페이지를 집중적으로 액세스하는 현상이다.
- 어느 하나의 페이지를 참조하면 그 근처의 페이지를 계속 참조할 가능성이 높음을 의미한다.
- **공간 구역성이 이루어지는 기억장소** : 배열 순회(Array Traversal, 배열 순례), 순차적 코드의 실행, 프로그래머들이 관련된 변수(데이터를 저장할 기억장소)들을 서로 근처에 선언하여 할당되는 기억장소, 같은 영역에 있는 변수를 참조할 때 사용

핵심 283 워킹 셋(Working Set)

워킹 셋은 프로세스가 일정 시간 동안 자주 참조하는 페이지들의 집합이다.

- 데닝(Denning)이 제안한 프로그램의 움직임에 대한 모델로, 프로그램의 Locality 특징을 이용한다.
- 자주 참조되는 워킹 셋을 주기억장치에 상주시킴으로써 페이지 부재 및 페이지 교체 현상이 줄어들어 프로세스의 기억장치 사용이 안정된다.
- 시간이 지남에 따라 자주 참조하는 페이지들의 집합이 변화하기 때문에 워킹 셋은 시간에 따라 변경된다.

핵심 284 스래싱(Thrashing)

스래싱은 프로세스의 처리 시간보다 페이지 교체에 소요되는 시간이 더 많아지는 현상이다.

- 다중 프로그래밍 시스템이나 가상기억장치를 사용하는 시스템에서 하나의 프로세스 수행 과정중 자주 페이지 부재가 발생함으로써 나타나는 현상으로, 전체 시스템의 성능이 저하된다.
- 다중 프로그래밍의 정도가 높아짐에 따라 CPU의 이용률은 어느 특정 시점까지는 높아지지만, 다중 프로그래밍의 정도가 더욱 커지면 스래싱이 나타나고, CPU의 이용률은 급격히 감소하게 된다.
- 스래싱 현상 방지 방법
  - 다중 프로그래밍의 정도를 적정 수준으로 유지한다.
  - 페이지 부재 빈도(Page Fault Frequency)를 조절하여 사용한다.
  - 워킹 셋을 유지한다.
  - 부족한 자원을 증설하고, 일부 프로세스를 중단시킨다.
  - CPU 성능에 대한 자료의 지속적 관리 및 분석으로 임계치를 예상하여 운영한다.

※ 페이지 부재(Page Fault)

- 페이지 부재는 프로세스 실행 시 참조할 페이지가 주기억장치에 없는 현상이며, 페이지 부재 빈도(PFF; Page Fault Frequency)는 페이지 부재가 일어나는 횟수를 의미한다.
- **페이지 부재 빈도 방식** : 페이지 부재율(Page Fault Rate)에 따라 주기억장치에 있는 페이지 프레임의 수를 늘리거나 줄여 페이지 부재율을 적정 수준으로 유지하는 방식

핵심 285 프로세스(Process)의 정의

프로세스는 일반적으로 프로세서(처리기, CPU)에 의해 처리되는 사용자 프로그램, 시스템 프로그램, 즉 실행중인 프로그램을 의미하며, 작업(Job), 태스크(Task)라고도 한다.

- 프로세스는 다음과 같이 여러 형태로 정의할 수 있다.
  - PCB를 가진 프로그램
  - 실기억장치에 저장된 프로그램
  - 프로세서가 할당되는 실체로서, 디스패치가 가능한 단위
  - 프로시저가 활동중인 것
  - 비동기적 행위를 일으키는 주체
  - 지정된 결과를 얻기 위한 일련의 계통적 동작
  - 목적 또는 결과에 따라 발생되는 사건들의 과정
  - 운영체제가 관리하는 실행 단위

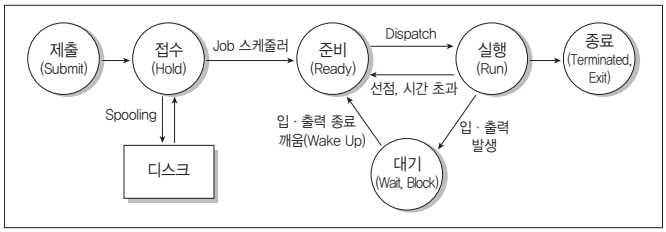
핵심 286 PCB

PCB(Process Control Block, 프로세스 제어 블록)는 운영체제가 프로세스에 대한 중요한 정보를 저장해 놓는 곳으로, Task Control Block 또는 Job Control Block이라고도 한다.

- 각 프로세스가 생성될 때마다 고유의 PCB가 생성되고, 프로세스가 완료되면 PCB는 제거된다.
- PCB에 저장되어 있는 정보
  - 프로세스의 현재 상태
  - 포인터
    - ▶ 부모 프로세스에 대한 포인터
    - ▶ 자식 프로세스에 대한 포인터
    - ▶ 프로세스가 위치한 메모리에 대한 포인터
    - ▶ 할당된 자원에 대한 포인터
  - 프로세스 고유 식별자
  - 스케줄링 및 프로세스의 우선순위
  - CPU 레지스터 정보
  - 주기억장치 관리 정보
  - 입 · 출력 상태 정보
  - 계정 정보

핵심 287 프로세스 상태 전이

프로세스 상태 전이는 프로세스가 시스템 내에 존재하는 동안 프로세스의 상태가 변하는 것을 의미하며, 프로세스의 상태를 다음과 같이 상태 전이도로 표시할 수 있다.



- 프로세스의 상태는 제출, 접수, 준비, 실행, 대기 상태로 나눌 수 있으며, 이 중 주요 세 가지 상태는 준비, 실행, 대기 상태이다.
- **제출(Submit)** : 작업을 처리하기 위해 사용자가 작업을 시스템에 제출한 상태이다.
- **접수(Hold)** : 제출된 작업이 스풀 공간인 디스크의 할당 위치에 저장된 상태이다.
- **준비(Ready)** : 프로세스가 프로세서를 할당받기 위해 기다리고 있는 상태이다.
- **실행(Run)** : 준비상태 큐에 있는 프로세스가 프로세서를 할당받아 실행되는 상태이다.
- **대기(Wait), 보류, 블록(Block)** : 프로세스에 입 · 출력 처리가 필요하면 현재 실행 중인 프로세스가 중단되고, 입 · 출력 처리가 완료될 때까지 대기하고 있는 상태이다.
- **종료(Terminated, Exit)** : 프로세스의 실행이 끝나고 프로세스 할당이 해제된 상태이다.

핵심 288 프로세스 상태 전이 관련 용어

Dispatch	준비 상태에서 대기하고 있는 프로세스 중 하나가 프로세서를 할당받아 실행 상태로 전이되는 과정이다.
Wake Up	입·출력 작업이 완료되어 프로세스가 대기 상태에서 준비 상태로 전이 되는 과정이다.
Spooling	입·출력장치의 공유 및 상대적으로 느린 입·출력장치의 처리 속도를 보완하고 다중 프로그래밍 시스템의 성능을 향상시키기 위해 입·출력할 데이터를 직접 입·출력장치에 보내지 않고 나중에 한꺼번에 입·출력하기 위해 디스크에 저장하는 과정이다.
교통량 제어기 (Traffic Controller)	프로세스의 상태에 대한 조사와 통보를 담당한다.

핵심 289 스레드(Thread)

스레드는 프로세스 내에서의 작업 단위로서 시스템의 여러 자원을 할당받아 실행하는 프로그램의 단위이다.

- 하나의 프로세스에 하나의 스레드가 존재하는 경우에는 단일 스레드, 하나 이상의 스레드가 존재하는 경우에는 다중 스레드라고 한다.
- 프로세스의 일부 특성을 갖고 있기 때문에 경량(Light Weight) 프로세스라고도 한다.
- 스레드 기반 시스템에서 스레드는 독립적인 스케줄링의 최소 단위로서 프로세스의 역할을 담당한다.
- 동일 프로세스 환경에서 서로 독립적인 다중 수행이 가능하다.
- 스레드의 분류

사용자 수준의 스레드	• 사용자가 만든 라이브러리를 사용하여 스레드를 운용한다. • 속도는 빠르지만 구현이 어렵다.
커널 수준의 스레드	• 운영체제의 커널에 의해 스레드를 운용한다. • 구현이 쉽지만 속도가 느리다

- 스레드 사용의 장점
  - 하나의 프로세스를 여러 개의 스레드로 생성하여 병행성을 증진시킬 수 있다.
  - 하드웨어, 운영체제의 성능과 응용 프로그램의 처리율을 향상시킬 수 있다.
  - 응용 프로그램의 응답 시간(Response Time)을 단축시킬 수 있다.
  - 실행 환경을 공유시켜 기억장소의 낭비가 줄어든다.
  - 프로세스들 간의 통신이 향상된다.
  - 스레드는 공통적으로 접근 가능한 기억장치를 통해 효율적으로 통신한다.

핵심 290 스케줄링 / 문맥 교환

스케줄링의 정의

스케줄링(Scheduling)은 프로세스가 생성되어 실행될 때 필요한 시스템의 여러 자원을 해당 프로세스에게 할당하는 작업을 의미한다.

스케줄링의 목적

- 공정성
- 처리율(량) 증가
- CPU 이용률 증가
- 우선순위 제도
- 오버헤드 최소화
- 응답 시간(Response Time, 반응 시간) 최소화
- 반환 시간(Turn Around Time) 최소화
- 대기 시간 최소화
- 균형 있는 자원의 사용
- 무한 연기 회피

문맥 교환

- 문맥 교환은 하나의 프로세스에서 다른 프로세스로 CPU가 할당되는 과정에서 발생되는 것이다.
- 새로운 프로세스에 CPU를 할당하기 위해 현재 CPU가 할당된 프로세스의 상태 정보를 저장한다.
- 새로운 프로세스의 상태 정보를 설정한 후 CPU를 할당하여 실행되도록 한다.

※ 스케줄링의 성능 평가 기준 : 스케줄링의 목적 중 CPU 이용률, 처리율, 반환 시간, 대기 시간, 응답 시간은 여러 종류의 스케줄링 성능을 비교하는 기준이 됨

핵심 291 프로세스 스케줄링의 기법

비선점(Non-Preemptive) 스케줄링

- 이미 할당된 CPU를 다른 프로세스가 강제로 빼앗아 사용할 수 없는 스케줄링 기법이다.
- 프로세스가 CPU를 할당받으면 해당 프로세스가 완료될 때까지 CPU를 사용한다.
- 모든 프로세스에 대한 요구를 공정하게 처리할 수 있다.
- 프로세스 응답 시간의 예측이 용이하며, 일괄 처리 방식에 적합하다.
- 중요한 작업(짧은 작업)이 중요하지 않은 작업(긴 작업)을 기다리는 경우가 발생할 수 있다.
- 비선점 스케줄링의 종류에는 FCFS, SJF, 우선순위, HRN, 기한부 등의 알고리즘이 있다.

선점(Preemptive) 스케줄링

- 하나의 프로세스가 CPU를 할당받아 실행하고 있을 때 우선순위가 높은 다른 프로세스가 CPU를 강제로 빼앗아 사용할 수 있는 스케줄링 기법이다.
- 우선순위가 높은 프로세스를 빠르게 처리할 수 있다.
- 주로 빠른 응답 시간을 요구하는 대화식 시분할 시스템에 사용된다.
- 많은 오버헤드(Overhead)를 초래한다.
- 선점이 가능하도록 일정 시간 배당에 대한 인터럽트용 타이머 클록(Clock)이 필요하다.
- 선점 스케줄링의 종류에는 Round Robin, SRT, 선점 우선순위, 다단계 큐, 다단계 피드백 큐 등의 알고리즘이 있다.

핵심 292 주요 스케줄링 알고리즘

FCFS(First Come First Service, 선입 선출) = FIFO(First In First Out)

FCFS는 준비상태 큐(대기 큐, 준비 완료 리스트, 작업준비 큐, 스케줄링 큐)에 도착한 순서에 따라 차례로 CPU를 할당하는 기법으로, 가장 간단한 알고리즘이다.

- 먼저 도착한 것이 먼저 처리되어 공평성은 유지되지만 짧은 작업이 긴 작업을, 중요한 작업이 중요하지 않은 작업을 기다리게 된다.

SJF(Shortest Job First, 단기 작업 우선)

SJF는 준비상태 큐에서 기다리고 있는 프로세스들 중에서 실행 시간이 가장 짧은 프로세스에게 먼저 CPU를 할당하는 기법이다.

- 가장 적은 평균 대기 시간을 제공하는 최적 알고리즘이다.

HRN(Hightest Response-ratio Next)

실행 시간이 긴 프로세스에 불리한 SJF 기법을 보완하기 위한 것으로, 대기 시간과 서비스(실행) 시간을 이용하는 기법이다.

- 우선순위 계산 공식을 이용하여 서비스(실행) 시간이 짧은 프로세스나 대기 시간이 긴 프로세스에게 우선순위를 주어 CPU를 할당한다.
- 서비스 실행 시간이 짧거나 대기 시간이 긴 프로세스일 경우 우선순위가 높아진다.
- 우선순위를 계산하여 그 숫자가 가장 높은 것부터 낮은 순으로 우선순위가 부여된다.
- 우선순위 계산식

$$\text{우선순위 계산식} = \frac{\text{대기 시간+서비스 시간}}{\text{서비스 시간}}$$

핵심 293 Windows의 주요 환경 변수

Windows에서 환경 변수를 명령어나 스크립트에서 사용하려면 변수명 앞뒤에 '%'를 입력해야 한다.

- Windows에서 set을 입력하면 모든 환경 변수와 값을 출력한다.

환경 변수	용도
%ALLUSERPROFILE%	모든 사용자의 프로필이 저장된 폴더
%APPDATA%	설치된 프로그램의 필요 데이터가 저장된 폴더
%COMSPEC%	기본 명령 프롬프트로 사용할 프로그램명
%HOMEDRIVE%	로그인한 계정의 정보가 저장된 드라이브
%HOMEPATH%	로그인한 계정의 기본 폴더
%LOGONSERVER%	로그인한 계정이 접속한 서버명
%PATH%	실행 파일을 찾는 경로
%PATHEXT%	cmd에서 실행할 수 있는 파일의 확장자 목록
%PROGRAMFILES%	기본 프로그램의 설치 폴더
%SYSTEMDRIVE%	Windows가 부팅된 드라이브
%SYSTEMROOT%	부팅된 운영체제가 들어 있는 폴더
%TEMP% 또는 %TMP%	임시 파일이 저장되는 폴더
%USERDOMAIN%	로그인한 시스템의 도메인명
%USERNAME%	로그인한 계정 이름
%USERPROFILE%	로그인한 유저의 프로필이 저장된 폴더명

핵심 294 UNIX / LINUX의 주요 환경 변수

UNIX나 LINUX에서 환경 변수를 명령어나 스크립트에서 사용하려면 변수명 앞에 '$'를 입력해야 한다.

- UINIX나 LINUX에서는 set, env, printenv, setenv 중 하나를 입력하면 모든 환경 변수와 값을 표시한다.

환경 변수	용도
$DISPLAY	현재 X 윈도 디스플레이 위치
$HOME	사용자의 홈 디렉터리
$LANG	프로그램 사용 시 기본적으로 지원되는 언어
$MAIL	메일을 보관하는 경로
$PATH	실행 파일을 찾는 경로
$PS1	쉘 프롬프트 정보
$PWD	현재 작업하는 디렉터리
$TERM	로긴 터미널 타입
$USER	사용자의 이름

핵심 295 Windows 기본 명령어

명령어	기능
DIR	파일 목록을 표시한다.
COPY	파일을 복사한다.
TYPE	파일의 내용을 표시한다.
REN	파일의 이름을 변경한다.
DEL	파일을 삭제한다.
MD	디렉터리를 생성한다.
CD	디렉터리의 위치를 변경한다.
CLS	화면의 내용을 지운다.
ATTRIB	파일의 속성을 변경한다.
FIND	파일을 찾는다.
CHKDSK	디스크 상태를 점검한다.
FORMAT	디스크 표면을 트랙과 섹터로 나누어 초기화한다.
MOVE	파일을 이동한다.

핵심 296 UNIX / LINUX 기본 명령어

명령어	기능
cat	파일 내용을 화면에 표시한다.
chdir	현재 사용할 디렉터리의 위치를 변경한다.
chmod	파일의 보호 모드를 설정하여 파일의 사용 허가를 지정한다.
chown	소유자를 변경한다.
cp	파일을 복사한다.
exec	새로운 프로세스를 수행한다.
find	파일을 찾는다.
fork	새로운 프로세스를 생성한다(하위 프로세스 호출, 프로세스 복제 명령).
fsck	파일 시스템을 검사하고 보수한다.
getpid	자신의 프로세스 아이디를 얻는다.
getppid	부모 프로세스 아이디를 얻는다.
ls	현재 디렉터리 내의 파일 목록을 확인한다.
mount/ unmount	파일 시스템을 마운팅한다/마운팅 해제한다.
rm	파일을 삭제한다.
wait	fork 후 exec에 의해 실행되는 프로세스의 상위 프로세스가 하위 프로세스 종료 등의 event를 기다린다.

핵심 297 인터넷

IP 주소(Internet Protocol Address)

IP 주소는 인터넷에 연결된 모든 컴퓨터 자원을 구분하기 위한 고유한 주소이다.

- 숫자로 8비트씩 4부분, 총 32비트로 구성되어 있다.
- IP 주소는 네트워크 부분의 길이에 따라 다음과 같이 A 클래스에서 E 클래스까지 총 5단계로 구성되어 있다.

A Class	국가나 대형 통신망에 사용(0~127로 시작) 2^{24} = 16,777,216개의 호스트 사용 가능	1 8 9 16 17 24 25 32bit
B Class	중대형 통신망에 사용(128~191로 시작) 2^{16} = 65,536개의 호스트 사용 가능	
C Class	소규모 통신망에 사용(192~223으로 시작) 2^{8} = 256개의 호스트 사용 가능	
D Class	멀티캐스트용으로 사용(224~239로 시작)	네트워크 부분 호스트 부분
E Class	실험적 주소이며 공용되지 않음	

서브네팅(Subnetting)

서브네팅은 할당된 네트워크 주소를 다시 여러 개의 작은 네트워크로 나누어 사용하는 것을 말한다.

- 4바이트의 IP 주소 중 네트워크 주소와 호스트 주소를 구분하기 위한 비트를 서브넷 마스크(Subnet Mask)라고 하며, 이를 변경하여 네트워크 주소를 여러 개로 분할하여 사용한다.
- 서브넷 마스크는 각 클래스마다 다르게 사용된다.

IPv6(Internet Protocol version 6)

IPv6은 현재 사용하고 있는 IP 주소 체계인 IPv4의 주소 부족 문제를 해결하기 위해 개발되었다.

- 128비트의 긴 주소를 사용하여 주소 부족 문제를 해결할 수 있으며, IPv4에 비해 자료 전송 속도가 빠르다.
- 인증성, 기밀성, 데이터 무결성의 지원으로 보안 문제를 해결할 수 있다.
- IPv4와 호환성이 뛰어나다.
- 주소의 확장성, 융통성, 연동성이 뛰어나며, 실시간 흐름 제어로 향상된 멀티미디어 기능을 지원한다.
- 패킷 크기를 확장할 수 있으므로 패킷 크기에 제한이 없다.

IPv6의 구성

- 16비트씩 8부분, 총 128비트로 구성되어 있다.
- 각 부분을 16진수로 표현하고, 콜론(:)으로 구분한다.
- IPv6은 다음과 같이 세 가지 주소 체계로 나누어진다.

유니캐스트 (Unicast)	단일 송신자와 단일 수신자 간의 통신(1 대 1 통신에 사용)
멀티캐스트 (Multicast)	단일 송신자와 다중 수신자 간의 통신(1 대 다 통신에 사용)
애니캐스트 (Anycast)	단일 송신자와 가장 가까이 있는 단일 수신자 간의 통신(1 대 1 통신에 사용)

도메인 네임(Domain Name)

도메인 네임은 숫자로 된 IP 주소를 사람이 이해하기 쉬운 문자 형태로 표현한 것이다.

- 호스트 컴퓨터 이름, 소속 기관 이름, 소속 기관의 종류, 소속 국가명 순으로 구성되며, 왼쪽에서 오른쪽으로 갈수록 상위 도메인을 의미한다.

핵심 298 OSI 참조 모델

- 다른 시스템 간의 원활한 통신을 위해 ISO(국제표준화기구)에서 제안한 통신 규약(Protocol)이다.
- OSI 7계층은 1~3 계층을 하위 계층, 4~7 계층을 상위 계층이라고 한다.
  - 하위 계층 : 물리 계층 → 데이터 링크 계층 → 네트워크 계층
  - 상위 계층 : 전송 계층 → 세션 계층 → 표현 계층 → 응용 계층

물리 계층 (Physical Layer)	전송에 필요한 두 장치 간의 실제 접속과 절단 등 기계적, 전기적, 기능적, 절차적 특성에 대한 규칙을 정의한다.
데이터 링크 계층 (Data Link Layer)	• 두 개의 인접한 개방 시스템들 간에 신뢰성 있고 효율적인 정보 전송을 할 수 있도록 시스템 간 연결 설정과 유지 및 종료를 담당한다. • 송신 측과 수신 측의 속도 차이를 해결하기 위한 흐름 제어 기능을 한다. • 프레임의 시작과 끝을 구분하기 위한 프레임의 동기화 기능을 한다. • 오류의 검출과 회복을 위한 오류 제어 기능을 한다.
네트워크 계층 (Network Layer, 망 계층)	• 개방 시스템들 간의 네트워크 연결을 관리하는 기능과 데이터의 교환 및 중계 기능을 한다. • 네트워크 연결을 설정, 유지, 해제하는 기능을 한다. • 경로 설정(Routing), 데이터 교환 및 중계, 트래픽 제어, 패킷 정보 전송을 수행한다.
전송 계층 (Transport Layer)	• 논리적 안정과 균일한 데이터 전송 서비스를 제공함으로써 종단 시스템(End-to-End) 간에 투명한 데이터 전송을 가능하게 한다. • 종단 시스템(End-to-End) 간의 전송 연결 설정, 데이터 전송, 연결 해제 기능을 한다. • 주소 설정, 다중화(분할 및 재조립), 오류 제어, 흐름 제어를 수행한다.
세션 계층 (Session Layer)	• 송 · 수신 측 간의 관련성을 유지하고 대화 제어를 담당한다. • 대화(회화) 구성 및 동기 제어, 데이터 교환 관리 기능을 한다.
표현 계층 (Presentation Layer)	• 응용 계층으로부터 받은 데이터를 세션 계층에 보내기 전에 통신에 적당한 형태로 변환하고, 세션 계층에서 받은 데이터는 응용 계층에 맞게 변환하는 기능을 한다. • 서로 다른 데이터 표현 형태를 갖는 시스템 간의 상호 접속을 위해 필요한 계층이다. • 코드 변환, 데이터 암호화, 데이터 압축, 구문 검색, 정보 형식(포맷) 변환, 문맥 관리 기능을 한다.
응용 계층 (Application Layer)	사용자(응용 프로그램)가 OSI 환경에 접근할 수 있도록 서비스를 제공한다.

핵심 299 네트워크 관련 장비

네트워크 인터페이스 카드 (NIC; Network Interface Card)	컴퓨터와 컴퓨터 또는 컴퓨터와 네트워크를 연결하는 장치로, 정보 전송 시 정보가 케이블을 통해 전송될 수 있도록 정보 형태를 변경한다.
허브(Hub)	• 한 사무실이나 가까운 거리의 컴퓨터들을 연결하는 장치로, 각 회선을 통합적으로 관리하며, 신호 증폭 기능을 하는 리피터의 역할도 포함한다. • 허브의 종류에는 더미 허브, 스위칭 허브가 있다.
리피터(Repeater)	전송되는 신호가 전송 선로의 특성 및 외부 충격 등의 요인으로 인해 원래의 형태와 다르게 왜곡되거나 약해질 경우 원래의 신호 형태로 재생하여 다시 전송하는 역할을 수행한다.
브리지(Bridge)	• LAN과 LAN을 연결하거나 LAN 안에서의 컴퓨터 그룹(세그먼트)을 연결하는 기능을 수행한다. • 네트워크를 분산적으로 구성할 수 있어 보안성을 높일 수 있다.

스위치(Switch)	• 브리지와 같이 LAN과 LAN을 연결하여 훨씬 더 큰 LAN을 만드는 장치이다. • 하드웨어를 기반으로 처리하므로 전송 속도가 빠르다.
라우터(Router)	브리지와 같이 LAN과 LAN의 연결 기능에 데이터 전송의 최적 경로를 선택할 수 있는 기능이 추가된 것으로, 서로 다른 LAN이나 LAN과 WAN의 연결도 수행한다.
게이트웨이(Gateway)	• 전 계층(1~7계층)의 프로토콜 구조가 다른 네트워크의 연결을 수행한다. • LAN에서 다른 네트워크에 데이터를 보내거나 다른 네트워크로부터 데이터를 받아들이는 출입구 역할을 한다.

핵심 300 프로토콜의 개념

- 프로토콜은 서로 다른 기기들 간의 데이터 교환을 원활하게 수행할 수 있도록 표준화시켜 놓은 통신 규약이다.
- 프로토콜의 기본 요소
  - 구문(Syntax) : 전송하고자 하는 데이터의 형식, 부호화, 신호 레벨 등을 규정
  - 의미(Semantics) : 두 기기 간의 효율적이고 정확한 정보 전송을 위한 협조 사항과 오류 관리를 위한 제어 정보를 규정
  - 시간(Timing) : 두 기기 간의 통신 속도, 메시지의 순서 제어 등을 규정
- 프로토콜의 기능

단편화와 재결합	송신 측에서 전송할 데이터를 전송에 알맞은 일정 크기의 작은 블록으로 자르는 작업을 단편화(Fragmentation)라 하고, 수신 측에서 단편화된 블록을 원래의 데이터로 모으는 것을 재결합(Reassembly)이라 한다.
캡슐화	캡슐화(Encapsulation)는 단편화된 데이터에 송 · 수신지 주소, 오류 검출 코드, 프로토콜 기능을 구현하기 위한 프로토콜 제어 정보 등의 정보를 부가하는 것으로, 요약화라고도 한다.
흐름 제어	• 흐름 제어(Flow Control)는 수신 측의 처리 능력에 따라 송신 측에서 송신하는 데이터의 전송량이나 전송 속도를 조절하는 기능이다. • 정지-대기(Stop-and-Wait), 슬라이딩 윈도우(Sliding Window) 방식을 이용한다.
오류 제어	오류 제어(Error Control)는 전송중에 발생하는 오류를 검출하고 정정하여 데이터나 제어 정보의 파손에 대비하는 기능이다.
동기화	동기화(Synchronization)는 송 · 수신 측이 같은 상태를 유지하도록 타이밍(Timing)을 맞추는 기능이다.
순서 제어	순서 제어(Sequencing)는 전송되는 데이터 블록(PDU)에 전송 순서를 부여하는 기능으로, 연결 위주의 데이터 전송 방식에만 사용된다.
주소 지정	주소 지정(Addressing)은 데이터가 목적지까지 정확하게 전송될 수 있도록 목적지 이름, 주소, 경로를 부여하는 기능이다.
다중화	다중화(Multiplexing)는 한 개의 통신 회선을 여러 가입자들이 동시에 사용하도록 하는 기능이다.
경로 제어	경로 제어(Routing)는 송 · 수신 측 간의 송신 경로 중에서 최적의 패킷 교환 경로를 설정하는 기능이다.
전송 서비스	전송하려는 데이터가 사용하도록 하는 별도의 부가 서비스이다.

핵심 301 TCP/IP

- TCP/IP는 인터넷에 연결된 서로 다른 기종의 컴퓨터들이 데이터를 주고받을 수 있도록 하는 표준 프로토콜이다.
- TCP/IP는 TCP 프로토콜과 IP 프로토콜이 결합된 것을 의미한다.
- TCP/IP는 응용 계층, 전송 계층, 인터넷 계층, 네트워크 액세스 계층으로 이루어져 있다.
- 응용 계층의 주요 프로토콜

FTP (File Transfer Protocol)	컴퓨터와 컴퓨터 또는 컴퓨터와 인터넷 사이에서 파일을 주고받을 수 있도록 하는 원격 파일 전송 프로토콜
SMTP(Simple Mail Transfer Protocol)	전자 우편을 교환하는 서비스
TELNET	• 멀리 떨어져 있는 컴퓨터에 접속하여 자신의 컴퓨터처럼 사용할 수 있도록 해주는 서비스 • 프로그램을 실행하는 등 시스템 관리 작업을 할 수 있는 가상의 터미널(Virtual Terminal) 기능을 수행

SNMP (Simple Network Management Protocol)	TCP/IP의 네트워크 관리 프로토콜로, 라우터나 허브 등 네트워크 기기의 네트워크 정보를 네트워크 관리 시스템에 보내는 데 사용되는 표준 통신 규약
DNS (Domain Name System)	도메인 네임을 IP 주소로 매핑(Mapping)하는 시스템
HTTP(HyperText Transfer Protocol)	월드 와이드 웹(WWW)에서 HTML 문서를 송수신 하기 위한 표준 프로토콜

• 전송 계층의 주요 프로토콜

TCP (Transmission Control Protocol)	• 양방향 연결(Full Duplex Connection)형 서비스를 제공한다. • 스트림 위주의 전달(패킷 단위)을 한다. • 신뢰성 있는 경로를 확립하고 메시지 전송을 감독한다. • 순서 제어, 오류 제어, 흐름 제어 기능을 한다. • TCP 프로토콜의 헤더는 기본적으로 20Byte에서 60Byte까지 사용할 수 있는데, 선택적으로 40Byte를 더 추가할 수 있으므로 최대 100Byte까지 크기를 확장할 수 있다.
UDP (User Datagram Protocol)	• 데이터 전송 전에 연결을 설정하지 않는 비연결형 서비스를 제공한다. • TCP에 비해 상대적으로 단순한 헤더 구조를 가지므로, 오버헤드가 적고, 흐름제어나 순서 제어가 없어 전송 속도가 빠르다. • 실시간 전송에 유리하며, 신뢰성보다는 속도가 중요시되는 네트워크에서 사용된다.
RTCP (Real-Time Control Protocol)	• RTP(Real-time Transport Protocol) 패킷의 전송 품질을 제어하기 위한 제어 프로토콜이다. • 세션(Session)에 참여한 각 참여자들에게 주기적으로 제어 정보를 전송한다.

• 인터넷 계층의 주요 프로토콜

IP(Internet Protocol)	• 전송할 데이터에 주소를 지정하고, 경로를 설정하는 기능을 한다. • 비연결형인 데이터그램 방식을 사용하는 것으로 신뢰성이 보장되지 않는다.
ICMP (Internet Control Message Protocol, 인터넷 제어 메시지 프로토콜)	IP와 조합하여 통신중에 발생하는 오류의 처리와 전송 경로 변경 등을 위한 제어 메시지를 관리하는 역할을 하며, 헤더는 8Byte로 구성된다.
IGMP(Internet Group Management Protocol, 인터넷 그룹 관리 프로토콜)	멀티캐스트를 지원하는 호스트나 라우터 사이에서 멀티캐스트 그룹 유지를 위해 사용된다.
ARP (Address Resolution Protocol, 주소 분석 프로토콜)	호스트의 IP 주소를 호스트와 연결된 네트워크 접속 장치의 물리적 주소(MAC Address)로 바꾼다.
RARP (Reverse Address Resolution Protocol)	ARP와 반대로 물리적 주소를 IP 주소로 변환하는 기능을 한다.

• 네트워크 액세스 계층의 주요 프로토콜

Ethernet (IEEE 802.3)	CSMA/CD 방식의 LAN
IEEE 802	LAN을 위한 표준 프로토콜
HDLC	비트 위주의 데이터 링크 제어 프로토콜
X.25	패킷 교환망을 통한 DTE와 DCE 간의 인터페이스를 제공하는 프로토콜
RS-232C	공중 전화 교환망(PSTN)을 통한 DTE와 DCE 간의 인터페이스를 제공하는 프로토콜